PERANCANGAN
SISTEM PENDUKUNG KEPUTUSAN KRITERIA MAJEMUK PADA PENGAMBILAN KEPUTUSAN KELOMPOK

DISERTASI

Karya Tulis sebagai salah satu syarat
untuk memperoleh gelar Doktor dari
Institut Teknologi Bandung

Oleh
MUHAMMAD ALI RAMDHANI
NIM 322 97 009

INSTITUT TEKNOLOGI BANDUNG
2001

ABSTRAK DISERTASI

MUHAMMAD ALI RAMDHANI

INSTITUT TEKNOLOGI BANDUNG
2001

PERANCANGAN
SISTEM PENDUKUNG KEPUTUSAN KRITERIA MAJEMUK PADA PENGAMBILAN KEPUTUSAN KELOMPOK

ABSTRAK DISERTASI

Karya Tulis sebagai salah satu syarat
untuk memperoleh gelar Doktor dari
Institut Teknologi Bandung

Oleh
MUHAMMAD ALI RAMDHANI

Promotor : 1. Prof. Ir. Harsono Taroepratjeka, Ph.D.
2. Prof. Dr. Ir. Iman Sudirman
3. Dr. Ir. Kadarsah Suryadi

INSTITUT TEKNOLOGI BANDUNG
2001

ABSTRAK DISERTASI

Disertasi ini memuat konsep pengembangan model dan perancangan Sistem Pendukung Keputusan Kriteria Majemuk (SPKKM) untuk pengambilan keputusan kelompok sebagai instrumen penetapan prioritas keputusan melalui pendekatan Pengambilan Keputusan Kriteria Majemuk (PKKM).

Sebuah model pengambilan keputusan merupakan instrumen metodologik yang bersyaratkan rasional yang bertujuan untuk mengurangi resiko dan meningkatkan nilai utilitas berdasarkan data yang diperoleh. Sistematika dari perancangan model dimulai dengan penyusunan model keputusan yang dilakukan dengan cara mengembangkan hubungan-hubungan logis yang mendasari persoalan keputusan (model konseptual), kemudian dituangkan ke dalam suatu model matematis, yang mencerminkan hubungan yang terjadi diantara faktor-faktor yang terlibat.

Pengembangan model *Analytic Hierarchy Process* (AHP) pada penelitian ini, menghasilkan enam prinsip dalam penyelesaian masalah persoalan PKKM, yaitu: dekomposisi, penilaian perbandingan, analisis interdependensi, sintesa prioritas, pembentukkan konsensus, dan penetapan alternatif.

Inti model yang dikembangkan pada penelitian ini adalah merinci suatu keadaan yang kompleks atau tak berkerangka ke dalam komponen-komponennya, kemudian mengatur bagian-bagian dari komponen-komponen tersebut dalam bentuk hirarki (dekomposisi), memberikan bobot verbal antar elemen dan menghitung bobot numerik (penilaian perbandingan), dan analisis interdependensi pada variabel yang dianggap penting (analisis interdependensi), dan selanjutnya melakukan sintesa dari pendapat tadi untuk menentukan variabel alternatif yang memiliki prioritas tertinggi (sintesa prioritas). Pada bagian akhir analisa keputusan dilakukan proses pengendalian output dengan menggunakan model programa linier, dengan input berupa nilai dan variabel kendala (*constraints*) yang akan dihadapi sebagai implikasi

implementasi alternatif (penetapan alternatif). Input data untuk programa linier merupakan bentuk survai (penelusuran fakta) dalam ukuran yang operasional.

Penelitian ini mengajukan model konsensus untuk pengambilan keputusan kelompok. Metoda konsensus tersebut dapat memberikan informasi mengenai tingkat kesepakatan dan ketidaksepakatan, serta rentang ketidaksepakatan individu dan kelompok, struktur klaster, mengidentifikansi objek keputusan "bermasalah" dan pendapat marjinal. Pembentukkan konsesus dilakukan berdasarkan pendekatan metoda Delphi, diskusi (tukar menukar informasi) dilakukan untuk memperoleh pendapat yang homogen. Aturan penghentian diskusi dilakukan ketika dijumpai pendapat homogen yang lebih dari 66.6 persen, atau berdasarkan keterbatasan waktu.

Untuk memudahkan penggunaan model ini, maka dirancang suatu SPKKM sebagai sistem yang dibangun dari model yang dikembangkan. Metodologi rancang bangun SPKKM menggunakan pendekatan *top-down* yang dititikberatkan pada pertimbangan kemampuan *Representation, Operation, Memory,* dan *Control Mechanism* (pendekatan *ROMC*) komponen SPK untuk kemudian diintegrasikan menjadi suatu sistem.

SPKKM dirancang untuk memanfaatkan analisis ilmiah pada permasalahan pengambilan keputusan, dengan tujuan untuk pengembangan dan pengelolaan sistem operasi, dan perancangan sistem informasi untuk pengambilan keputusan. Prinsip dasar bangunan sistem adalah bahwa setiap sistem diikat bersama oleh pertukaran informasi.

Validasi SPKKM diukur berdasarkan tingkat kemanfaatan (*usefulness*), kemudahan penggunaan (*usability*), kemampuan representasi dari situasi permasalahan, dan pertimbangan biaya, serta variabel penting lainnya. Kriteria untuk melihat validitas sistem didasarkan dan diukur pada pertimbangan filosofi keilmuan, yang terdiri dari *rasionalisme* dan e*mpirisme*.

DISSERTATION ABSTRACT

This dissertation discusses the concept in the development of Multi Criteria Decision Support System (MSDSS) model and design for group decision making as a means to determine decision priorities through the Multi Criteria Decision Making (MCDM) model.

A decision-making model is a methodological instrument that requires a rationale that is aimed at reducing risks and increasing utility values based on the data obtained. The systemization of model designs starts by compiling decision models that is carried out by means of developing logical relations that form the basis for the decision problem (conceptual model) to be later generated into a mathematical model which reflects the relations among the factors involved.

The development of *Analytic Hierarchy Process* (AHP) models in this research has resulted in six principles in settling Multi Criteria Decision Making problems, namely decomposition, comparative judgment, interdependence analysis, synthesis of priorities, consensus-building analysis and determining alternatives.

The core of the model developed in this research is to break down a complex and unstructured condition into detailed components, and later to arrange parts of the components into a hierarchical model (decomposition), giving verbal weights among the elements and calculating the numeric weights (comparative judgment) and interdependence analysis on variables that are deemed significant (interdependence analysis), and then synthesize the ideas to determine the alternative variable with the highest priority (synthesis of priorities). At the conclusion of the decision analysis, an output control process is carried out by using a linear programming model with input in the form of constraint values and variables, which are to be faced as implication of alternative implementation (determining alternative). The data input for linear programming is the form of survey (fact-finding) within operational scope.

This research puts forward a consensus model for group decision-making. The consensus method offers information concerning the level of agreement and the level of disagreement, and the range of individual and group agreement, structure of clusters; it also identifies the "problematic" decision object and the marginal opinions. Consensus is reached on the basis of the Delphi method approach, and discussion (information exchange) is carried out to obtain homogenous opinions. Discussion is terminated when homogenous opinions constitute over 66.6 percent, or due to time limit.

To facilitate the use of this model, a MCDSS is designed as a system built up of developed models. The design methodology of MCDSS applies the top-down approach, focused on considerations of capability of Representation, Operation and Control Mechanism (ROMC approach) of DSS components to be later integrated into one system.

MCDSS is designed to benefit from scientific analyses in solving decision-making problems, with the purpose to develop and manage the operational systems as well as to design the information system for decision-making. The basic principle of system design is that each system is bound to one another by exchange of information.

The validity of MCDSS is measured on the basis of the level of usefulness, usability, capability of representation of a problematic situation, consideration of benefit-cost ratio and other important variables. The criterion of the validity of the system is based on and measured by consideration of scientific philosophy, which comprises *rationalism* and *empiricism*.

PERANCANGAN
SISTEM PENDUKUNG KEPUTUSAN KRITERIA MAJEMUK
PADA PENGAMBILAN KEPUTUSAN KELOMPOK

MUHAMMAD ALI RAMDHANI

INSTITUT TEKNOLOGI BANDUNG
2001

PERANCANGAN
SISTEM PENDUKUNG KEPUTUSAN KRITERIA MAJEMUK PADA PENGAMBILAN KEPUTUSAN KELOMPOK

Oleh
Muhammad Ali Ramdhani

Menyetujui
Tim Promotor
Tanggal 29 Agustus 2001

Ketua,

(Prof. Ir. Harsono Taroepratjeka, Ph.D.)

Anggota,

(Prof. Dr. Ir. Iman Sudirman)

Anggota,

(Dr. Ir. Kadarsah Suryadi)

PEDOMAN PENGGUNAAN DISERTASI

Disertasi Doktor yang tidak dipublikasikan terdaftar dan tersedia di Perpustakaan Institut Teknologi Bandung, dan terbuka untuk umum dengan ketentuan bahwa hak cipta ada pada pengarang. Referensi kepustakaan diperkenankan dicatat, tetapi pengutipan atau peringkasan hanya dapat dilakukan seizin pengarang dan harus disertai dengan kebiasaan ilmiah untuk menyebutkan sumbernya.

Memperbanyak atau menerbitkan sebagian atau seluruh disertasi haruslah seizin Direktur Program Pascasarjana, Institut Teknologi Bandung.

Perpustakaan yang meminjam disertasi ini untuk keperluan anggotanya harus mengisi nama dan tanda tangan peminjam dan tanggal pinjam.

Dipersembahkan kepada :

Ibuku, yang menorehkan arti kejujuran padaku

Ayahku, yang menegaskan arti kerja keras dan disiplin

Istriku, yang menjadi inspirasi dan motivasi hidupku

Anakku, yang mengenalkan tanggung jawab padaku

Kakak-kakakku, yang menjelaskan makna keteladanan padaku

Adik-adikku, yang mengilhami arti kasih sayang padaku

Guru-guruku, yang membekali dan memberi makna hidupku

KATA PENGANTAR

Alhamdulillah, segala puji hanyalah bagi Allah, Tuhan Pengatur semesta alam, yang Maha Pengasih lagi Maha Penyayang, Pemilik Hari Kemudian. Hanya atas perkenan, rahmat dan karunia-Nya, serta berkat bantuan semua pihak dan pribadi-pribadi yang mulia hati, disertasi dengan judul "*Perancangan Sistem Pendukung Keputusan Kriteria Majemuk pada Pengambilan Keputusan Kelompok"* ini dapat diselesaikan.

Disertasi ini merupakan karya tulis yang disusun berdasarkan serangkaian kegiatan penelitian penulis, yang diajukan untuk memenuhi salah satu syarat untuk menyelesaikan Program Pendidikan Doktor (Strata 3) pada Fakultas Pascasarjana, Institut Teknologi Bandung.

Pada kesempatan ini, sekalipun sulit bagi penulis untuk mengungkapkan satu persatu, namun tanpa mengurangi yang lain, tak terbendung getar-getar perasaan yang lahir dari lubuk hati yang paling dalam, untuk menyampaikan rasa terima kasih kepada mereka yang menjadi bagian dalam penyelesaian proses pembuatan disertasi ini. Secara khusus penulis memberikan penghargaan dan terima kasih yang sebesar-besarnya kepada :

1 Prof. Ir. Harsono Taroepratjeka, Ph.D., Prof. Dr. Ir. Iman Sudirman, dan Dr. Ir. Kadarsah Suryadi selaku tim pembimbing yang telah banyak memberikan motivasi, arahan, koreksi, dan evaluasi selama penelitian dan penulisan disertasi ini.

2 Prof. Dr. Matthias Aroef, Prof. Dr. Ir. Isa Setiasyah Toha, Prof. Dr. J. Winardi, S.E., Prof. Dr. Eriyatno, dan Dr. Ir. Iftikar Z. Sutalaksana selaku tim penguji dan/atau penyanggah yang telah memberikan arahan, dan evaluasi atas disertasi ini.

3 Kedua orang tua penulis (Prof. H. Cecep Syarifuddin dan Dr. Hj. Ummu Salamah, M.S.), istri (Hj. Hilda Ainis Syifa), anak (Muhammad Khalifa

Umana), saudara-saudara (A. Syakur Amien, Tinneke Hermina, Aji Abdul Wahid, Wati Susilawati, Irfan Nabhani, Ahmad Syarif Munawi, dan Hilmi Aulawi,).

4 Prof. Dr. Yuhara Sukra yang telah memberikan bimbingan, arahan dan motivasi belajar bagi penulis.

5 Civitas Akademika Sekolah Tinggi Teknologi Garut dan Keluarga Besar Pondok Pesantren Al-Musadaddiyah Garut, terima kasih atas kasih sayang, pengertian dan kerjasamanya selama ini.

6 Kepada Keluarga Besar Pondok Pesantren Al-Aulia Bogor, dan Al-Masthuriyah Sukabumi, terima kasih atas inspirasi, motivasi, dukungan, dan perhatian yang mendalam terhadap seluruh aktivitas penulis

7 Segenap Staf Pengajar Jurusan Teknik Industri, Institut Teknologi Bandung atas perhatian dan bekal ilmu yang diberikan pada penulis.

8 Segenap Pengelola Sekretariat Program Pascasarjana, Bidang Studi Teknik dan Manajemen Industri, terima kasih atas bantuan dan persahabatannya.

9 Segenap Pengelola Sekretariat Program Doktor, Institut Teknologi Bandung, atas bantuan dan persahabatannya, khususnya pada Pak Iwan, dan Ibu Henny.

10 Sahabat-sahabat seperjuangan, khususnya buat Ali Anwar Yusuf, Iwan Alamsyah, Tedi Setiadi, Zulnasri, dan M. Ridwan Setiawan atas bantuan dan dukungan moril maupun materil selama berinteraksi dengan penulis.

11 Rekan-rekan mahasiswa S-3 atas kerjasamanya selama ini.

12 Semua pihak yang telah membantu selama penelitian dan penulisan Disertasi ini.

Semoga Allah SWT senantiasa memberikan rahmat serta karunia-Nya kepada beliau-beliau dan memberikan pahala yang berlipat ganda. Sebagai penutup, penulis berharap semoga Disertasi ini akan memberikan manfaat, khususnya kepada penulis, serta kepada bangsa, negara dan umat yang penulis cintai. Amien.

Bandung, Agustus 2001

Dipersembahkan kepada :

Ibuku, yang menorehkan arti kejujuran padaku

Ayahku, yang menegaskan arti kerja keras dan disiplin

Istriku, yang menjadi inspirasi dan motivasi hidupku

Anakku, yang mengenalkan tanggung jawab padaku

Kakak-kakakku, yang menjelaskan makna keteladanan padaku

Adik-adikku, yang mengilhami arti kasih sayang padaku

Guru-guruku, yang membekali dan memberi makna hidupku

KATA PENGANTAR

Alhamdulillah, segala puji hanyalah bagi Allah, Tuhan Pengatur semesta alam, yang Maha Pengasih lagi Maha Penyayang, Pemilik Hari Kemudian. Hanya atas perkenan, rahmat dan karunia-Nya, serta berkat bantuan semua pihak dan pribadi-pribadi yang mulia hati, disertasi dengan judul "*Perancangan Sistem Pendukung Keputusan Kriteria Majemuk pada Pengambilan Keputusan Kelompok"* ini dapat diselesaikan.

Disertasi ini merupakan karya tulis yang disusun berdasarkan serangkaian kegiatan penelitian penulis, yang diajukan untuk memenuhi salah satu syarat untuk menyelesaikan Program Pendidikan Doktor (Strata 3) pada Fakultas Pascasarjana, Institut Teknologi Bandung.

Pada kesempatan ini, sekalipun sulit bagi penulis untuk mengungkapkan satu persatu, namun tanpa mengurangi yang lain, tak terbendung getar-getar perasaan yang lahir dari lubuk hati yang paling dalam, untuk menyampaikan rasa terima kasih kepada mereka yang menjadi bagian dalam penyelesaian proses pembuatan disertasi ini. Secara khusus penulis memberikan penghargaan dan terima kasih yang sebesar-besarnya kepada :

1 Prof. Ir. Harsono Taroepratjeka, Ph.D., Prof. Dr. Ir. Iman Sudirman, dan Dr. Ir. Kadarsah Suryadi selaku tim pembimbing yang telah banyak memberikan motivasi, arahan, koreksi, dan evaluasi selama penelitian dan penulisan disertasi ini.

2 Prof. Dr. Matthias Aroef, Prof. Dr. Ir. Isa Setiasyah Toha, Prof. Dr. J. Winardi, S.E., Prof. Dr. Eriyatno, dan Dr. Ir. Iftikar Z. Sutalaksana selaku tim penguji dan/atau penyanggah yang telah memberikan arahan, dan evaluasi atas disertasi ini.

3 Kedua orang tua penulis (Prof. H. Cecep Syarifuddin dan Dr. Hj. Ummu Salamah, M.S.), istri (Hj. Hilda Ainis Syifa), anak (Muhammad Khalifa

Umana), saudara-saudara (A. Syakur Amien, Tinneke Hermina, Aji Abdul Wahid, Wati Susilawati, Irfan Nabhani, Ahmad Syarif Munawi, dan Hilmi Aulawi,).

4 Prof. Dr. Yuhara Sukra yang telah memberikan bimbingan, arahan dan motivasi belajar bagi penulis.

5 Civitas Akademika Sekolah Tinggi Teknologi Garut dan Keluarga Besar Pondok Pesantren Al-Musadaddiyah Garut, terima kasih atas kasih sayang, pengertian dan kerjasamanya selama ini.

6 Kepada Keluarga Besar Pondok Pesantren Al-Aulia Bogor, dan Al-Masthuriyah Sukabumi, terima kasih atas inspirasi, motivasi, dukungan, dan perhatian yang mendalam terhadap seluruh aktivitas penulis

7 Segenap Staf Pengajar Jurusan Teknik Industri, Institut Teknologi Bandung atas perhatian dan bekal ilmu yang diberikan pada penulis.

8 Segenap Pengelola Sekretariat Program Pascasarjana, Bidang Studi Teknik dan Manajemen Industri, terima kasih atas bantuan dan persahabatannya.

9 Segenap Pengelola Sekretariat Program Doktor, Institut Teknologi Bandung, atas bantuan dan persahabatannya, khususnya pada Pak Iwan, dan Ibu Henny.

10 Sahabat-sahabat seperjuangan, khususnya buat Ali Anwar Yusuf, Iwan Alamsyah, Tedi Setiadi, Zulnasri, dan M. Ridwan Setiawan atas bantuan dan dukungan moril maupun materil selama berinteraksi dengan penulis.

11 Rekan-rekan mahasiswa S-3 atas kerjasamanya selama ini.

12 Semua pihak yang telah membantu selama penelitian dan penulisan Disertasi ini.

Semoga Allah SWT senantiasa memberikan rahmat serta karunia-Nya kepada beliau-beliau dan memberikan pahala yang berlipat ganda. Sebagai penutup, penulis berharap semoga Disertasi ini akan memberikan manfaat, khususnya kepada penulis, serta kepada bangsa, negara dan umat yang penulis cintai. Amien.

Bandung, Agustus 2001

DAFTAR ISI

DAFTAR LAMPIRAN

DAFTAR GAMBAR

GAMBAR Halaman

GAMBAR Halaman

DAFTAR TABEL

DAFTAR PERSAMAAN

PERSAMAAN Halaman

DAFTAR SINGKATAN DAN LAMBANG

Singkatan	N a m a	Pemakaian pertama kali pada halaman
ACRD	Techniques for Analyzing Consensus Relevant Data	105
AHP	Analytic Hierarchy Process	5
AIJ	Aggregation of Individual Judgment	103
AIP	Aggregation of Individual Priorities	103
ANP	Analytic Network Process	35
CI	Consistency Index	98
CIH	Consistency Index of Hierarchy	100
CRH	Consistency Ratio of Hierarchy	99
CPU	Central Processing Unit	185
DBMS	Data Base Management Subsystem	60
DSS	Decision Support System	
EJOR	European Journal of Operational Research	
ETOP	Environment and Threats of Profile	4
F1	Fleksibilitas tingkat pertama	73
F2	Fleksibilitas tingkat kedua	74
F3	Fleksibilitas tingkat ketiga	75
F4	Fleksibilitas tingkat keempat	77
GSAQ	Group Strong Agreement Quotient	108
GSDI	Group Strongest Disagreement Indicator	108
GSDQ	Group Strongest Disagreement Quitient	108
ICV	Individual Consensus Vektor	107
ISAQ	Individual Strong Agreement Quotient	108
ASDI	Individual Strongest Disagreement Indicator	108
ISDQ	Individual Strongest Disagreement Quitient	108

Singkatan	N a m a	Pemakaian pertama kali pada halaman
IM/PO	Ilmu Manajemen/Penelitian Operasional	17
MAUT	Multiple Attribute Utility Theory	34
MBMS	Model Basic Management System	61
PDE	Pengolahan Data Elektronik	18
PKKM	Pengambilan Keputusan Kriteria Majemuk	1
RC	Ratio Consistency	98
RATIONAL	Ranking and Optimization in Choosing Alternatives	146
RI	Random Index	98
RIH	Random Index of Hierarchy	100
ROMC	Representations, Operations, Memory Aids, Control Mechanisms	70
SAP	Strategic Advantage Profile	4
SIM	Sistem Informasi Manajemen	17
SPK	Sistem Pendukung Keputusan	1
SPKKM	Sistem Pendukung Keputusan Kriteria Majemuk	1
SWOT	Strengths, Weaknesses, Opportunities, and Threats	4
OR	Operational Research	5
UTA	Utilité Additive	35
VAHP	Vector Space Formulation of the Analytic Hierarchy Process	109

LAMBANG	N a m a	Pemakaian pertama kali pada halaman
a_i	alternatif ke-*i*	27
a_{ij}	nilai preferensi untuk elemen ke-*i* dibandingkan dengan elemen ke-*j*	92
A	matriks perbandingan antar elemen	91
B_i	bobot nyata elemen ke-*i*	125
$C(.)$	nilai preferensi untuk suatu elemen berdasarkan pertimbangan suatu kriteria	29
c_j	kriteria ke-*j*	30
$c_{ij}^{'}$	gabungan kriteria ke-*i* dan kriteria ke-*j*	124
CI	Consistency Index	98
CRH	Consistency Ratio of Hierarchy	99
e	epsilon	102
$\hat{E}_i$	koherensi elemen ke-*i*	135
$f(.)$	suatu fungsi nilai	32
I	matriks identitas	97
K_i	nilai batasan ke-j (kendala/*constrain*)	137
max	maksimasi suatu fungsi	29
N_{ij}	nilai implementasi ke-*i* untuk alternatif ke-*j* dengan pertimbangan K_i	137
O	matriks nol	97
$\hat{P}_i$	koherensi individu ke-*i*	134
$r_{C_iC_j}$	korelasi antara elemen kriteria-i dan kriteria-j	121
RI	Random Index	98
RC	Ratio Consistency	98
$U(a)$	fungsi utilitas untuk elemen a	27
u_i	utilitas elemen ke-*i*	27
v_i	bobot preferensi kelompok elemen ke-*i*	130

LAMBANG	N a m a	Pemakaian pertama kali pada halaman
w_i	bobot elemen ke-*i*	92
w_i^*	Bobot elemen selain elemen ke-*i*	122
W	Vektor bobot	95
Z	fungsi tujuan	29
λ	eigenvalue	94
α	nilai batas kesepakatan	105
β	selisih sudut preferensi individu dengan kelompok	131
δ	nilai batas ketidaksepakatan	105
γ	nilai batas keanggotaan klaster	104
μ_{a_i}	bobot kecenderungan alternatif	127
$\mu_{a_i}^P$	bobot kecenderungan alternatif ke-*i* setelah perubahan pada analisis sensitivitas	129
ω_i	bobot akhir sintesa elemen ke-*i*	125
$\overline{\omega}_i$	bobot global	127
ω_i^P	bobot kriteria setelah perubahan pada analisis sensitivitas	128
$\Psi(\lambda)$	polynomial minimum	102
$\Psi'(\lambda)$	nilai derivatif awal yang berhubungan dengan λ	102
$\sum$	penjumlahan simultan	32
$\cup$	gabungan (union)	122
$\cap$	irisan	122

BAB I
PENDAHULUAN

1 Latar Belakang Masalah

Fokus kajian pada penelitian ini adalah perancangan Sistem Pendukung Keputusan Kriteria Majemuk (SPKKM) untuk menunjang pengambilan keputusan kelompok berbasiskan model Pengambilan Keputusan Kriteria Majemuk (PKKM). Sistematika penelitian dimulai dengan pengembangan model PKKM, kemudian dilanjutkan dengan perancangan SPKKM.

Penelitian ini dilatarbelakangi oleh kenyataan bahwa baik pada organisasi yang bergerak di industri produksi maupun jasa, tidak lepas dari persoalan manajemen pada umumnya. Perubahan struktur pasar, produk, teknologi, dan yang lainnya terus terjadi. Berbagai perubahan ini membuat organisasi harus membuat keputusan yang tepat demi mengatasi berbagai resiko yang ditimbulkan oleh perubahan-perubahan tersebut (Gandibleux, 1999; dan Doumpos dan Zopounidis, 2001).

Salah satu pendekatan pemecahan masalah tersebut adalah dengan mengembangkan prioritas aktivitas organisasi. Dalam kaitan ini, Stoner dan Wankel (1993) menyatakan bahwa dalam suatu proses pengambilan keputusan, pengambil keputusan terlebih dahulu harus memikirkan segala tujuan dan tindakannya, yang biasanya didasarkan atas suatu metoda, rencana, atau logika tertentu.

Guna membantu pengambil keputusan dalam mempercepat dan mempermudah proses pengambilan keputusan, diperlukan suatu perangkat pengambilan keputusan dalam bentuk SPK (Sistem Pendukung Keputusan). SPK merupakan pengembangan lebih lanjut dari Sistem Informasi Manajemen Terkomputerisasi (*Computerized Management Information System*), yang

dirancang sedemikian rupa sehingga bersifat interaktif dengan pemakainya. Sifat interaktif ini dimaksudkan untuk memudahkan integrasi antar berbagai komponen dalam proses pengambilan keputusan, seperti prosedur, kebijakan, teknik analisis, serta pengalaman dan wawasan managerial guna membentuk suatu metoda keputusan yang bersifat fleksibel.

SPK berbasis seperangkat elemen yang saling berinteraksi, membentuk suatu prosedur dalam pencapaian suatu tujuan atau tujuan-tujuan bersama, dengan mengoperasikan data pada waktu rujukan tertentu untuk menghasilkan informasi, yang kemudian digunakan sebagai referensi dalam pengambilan keputusan.

2 Perumusan Masalah

Persoalan pengambilan keputusan pada dunia nyata sangat sulit untuk memperoleh keputusan yang baik, apabila diselesaikan hanya dengan menggunakan pendekatan analisis kriteria tunggal. Untuk lebih realistis perlu dipertimbangkan seluruh aspek (kriteria majemuk) yang berkaitan dengan keputusan. Atau dengan kata lain, persoalan pada dunia nyata akan lebih baik apabila diselesaikan melalui pendekatan PKKM.

Perhatian utama dari PKKM adalah kerangka metodologi dalam melakukan analisa dan pemodelan pada preferensi pengambil keputusan. Oleh karena itu, pengembangan dari model keputusan kriteria majemuk dilakukan dengan pendekatan interaktif dan proses iteratif, hingga preferensi pengambil keputusan dapat direpresentasikan dalam model (Zopounidis dan Doumpos, 2000).

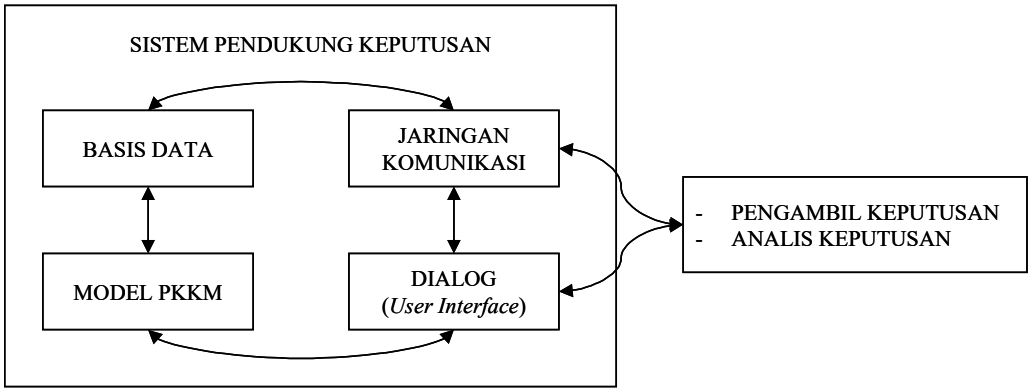
Interaktif dan proses iteratif merupakan dua karakteristik kunci dari SPK. Konsekuensinya, SPK bersama PKKM sebagai SPKKM menyediakan dukungan esensial dalam hal pembuatan struktur keputusan dan analisis perferensi dari pengambil keputusan. Pernyataan ini didukung oleh pernyataan

Espinasse et. al. (1997) yang menyatakan bahwa pendekatan PKKM adalah model yang relevan untuk dirancang dalam SPK, implementasi dari model dan teknik PKKM membantu dalam setiap tahapan perancangan analisa keputusan (penentuan alternatif) dan pemilihan alternatif.

Perancangan SPKKM ditujukan guna membantu pengambil keputusan dalam mempercepat dan mempermudah proses pengambilan keputusan. Tujuannya adalah untuk membantu pengambil keputusan memilih berbagai alternatif keputusan yang merupakan hasil pengolahan terhadap informasi-informasi yang diperoleh/tersedia dengan menggunakan model PKKM.

Analisis PKKM dengan menggunakan SPKKM meliputi kajian atas model, metoda dan pendekatan dengan tujuan untuk memberikan bantuan bagi pengambil keputusan untuk menangani persoalan keputusan semi terstruktur melalui pendekatan PKKM (Siskos dan Spyridakos, 1999).

Pengambil keputusan memperoleh dukungan dari SPKKM dalam pembuatan struktur masalah dengan mempertimbangkan seluruh kriteria, dan memungkinkan pengambil keputusan untuk membuat analisis konflik. Semua kapabilitas yang disediakan SPKKM memudahkan pengambil keputusan untuk melakukan proses pembelajaran dalam menangani persoalan secara lebih realistik dan memiliki basis argumentasi yang lebih utuh dalam menganalisa keputusan. SPKKM memungkinkan pengambil keputusan melihat secara jelas berbagai preferensi yang dikemukakan oleh para responden/pakar. Melalui analisis sensitivitas, pengambil keputusan dimungkinkan untuk menguji sejumlah skenario dengan melihat signifikansi suatu kriteria dan/atau perubahan lingkungan keputusan.



Gambar I.1 Struktur Hubungan antara SPK, Model PKKM, Analis Keputusan dan Pengambil Keputusan

3 Posisi Keilmuan (*The State of the Art*)

Model Pengambilan Keputusan Kriteria Majemuk

Beberapa persoalan pengambilan keputusan pada dunia nyata menetapkan inti permasalahannya pada penetapan urutan prioritas (*ranking, sorting*), hal ini hampir terjadi pada semua bidang, baik pada bidang keuangan, pemasaran, lingkungan, dan bahkan pada dunia kesehatan. Untuk beberapa dekade, permasalahan penetapan urutan prioritas dalam memilih di antara dua atau lebih alternatif menggunakan pendekatan statistik multivariat. Pada saat ini, kemungkinan pendekatan baru seperti sistem pakar *(expert systems), neural network*, programa matematik, PKKM, dan lain-lain telah diekploitasi dengan kerangka pengambilan keputusan yang lebih fleksibel.

Penelitian ini melakukan pengembangan model PKKM untuk mengatasi permasalahan penetapan prioritas/pemilihan alternatif. Pengembangan model PKKM ini dipandang perlu, mengingat beberapa organisasi masih menggunakan model konseptual, seperti SWOT (*Strengths, Weaknesses, Opportunities, and Threats*), SAP (*Strategic Advantage of Profile*) dan ETOP (*Environment and Threats of Profile*), padahal disadari bahwa model konseptual terkadang terlalu luas dan belum operasional, oleh karenanya model-model ini masih memerlukan proses idealisasi ke dalam suatu formulasi

model untuk kemudian dilakukan simbolisasi dan penetapan aturan kuantitatifnya.

Pada sisi lain, pada beberapa organisasi, banyak dijumpai penggunaan OR (*Operational Research*) yang klasik pada proses pengambilan keputusannya, padahal OR klasik belum tentu cocok digunakan untuk persoalan-persoalan yang kompleks. Pernyataan ini didukung oleh pendapat Putro dan Tjakraatmadja (1998) yang menyatakan bahwa OR klasik tidak mampu memecahkan persoalan pengambilan keputusan yang mengandung konflik, terutama karena OR mengasumsikan bahwa hanya ada satu tujuan (biasanya tujuan tersebut adalah *given*) yang akan dimaksimasi pada suatu kondisi yang pasti (*certainty*) atau suatu kondisi yang mengandung resiko dengan peluang yang diketahui. Di dalam kondisi konflik, upaya maksimasi oleh satu pihak akan dilawan oleh usaha minimasi di pihak lainnya, karena ada perbedaan preferensi. Walaupun OR menyediakan alat bantu untuk pemecahan konflik, yang disebut teori permainan (*game theory*), tetapi alat ini sangat terbatas pemakaiannya, karena banyak asumsi yang harus dipenuhi.

Pengembangan model yang dilakukan pada penelitian ini berbasis model *Analytic Hierarchy Process* (AHP) untuk pengambilan keputusan individu dan kelompok. Pengembangan model ini, dimaksudkan untuk memberikan kontribusi keilmuan dalam pengembangan model AHP berdasarkan suatu sudut pandang tertentu.

Pengembangan model yang dilakukan pada penelitian ini berfokus pada analisis **interdependensi kriteria** dan analisis **pembentukkan konsensus** partisipan AHP, dan **pemilihan alternatif**. Setelah sintesa prioritas, pada tahap akhir analisa dilakukan pemilihan alternatif yang paling layak untuk diimplementasikan melalui teknik programa linier (*linear programming*). Sehingga prinsip-prinsip yang digunakan dalam pemecahan masalahnya, hasil pengembangan AHP pada penelitian ini menganut prinsip-prinsip:

dekomposisi, **perbandingan penilaian**, **penilaian interdependensi kriteria**, **pembentukkan konsensus**, dan **pemilihan alternatif**.

Sistem Pendukung Keputusan Kriteria Majemuk

Konsep SPK pertama kali diperkenalkan oleh Keen dan Stanbell dengan pernyataan bahwa konsep dari pendukung keputusan melibatkan dua wilayah penelitian yang utama: pembelajaran teori dari pengorganisasian pengambilan keputusan yang dikerjakan di *Carniege Institute of Technology* pada akhir tahun 1950-an dan awal 1960-an, dan teknik kerja pada sistem interaktif yang diselesaikan di *Massachutes Institute of Technology* pada tahun 1960-an (Power, 1999).

Selanjutnya pada tahun 1971, konsep SPK dinyatakan Michael S. Scott Morton dengan istilah *Management Decision System*. Konsep SPK ditandai dengan sistem interaktif berbasis komputer yang membantu pengambil keputusan dengan memanfaatkan data dan model (Sprague dan Carlson, 1982). Penelitian Morton merupakan prinsip awal bagi implementasi, definisi, dan metoda penelitian untuk SPK khusus.

Pada tahun 1974, Gordon Davis di University of Minnesota mempublikasikan buku berjudul *Management Information System; Conceptual Foundations, Structure, and Development*. Pada Bab 12 bukunya diberi nama "*Information System Support for Decision Making*", dan pada Bab 13 diberi nama "*Information System Support for Planning and Control*" dibuat untuk pengembangan dasar penelitian SPK dan pembuatan SPK untuk kepentingan praktis.

Selanjutnya, tahun 1975 J. D. C. Little memperluas batasan komputer untuk mendukung pemodelan. SPK-nya dinamakan *Brandaid* dirancang untuk mendukung keputusan bidang produk, promosi, penghargaan, dan keputusan *advertising*. Little dalam artikel *Managemet Science* membuat artikel dengan

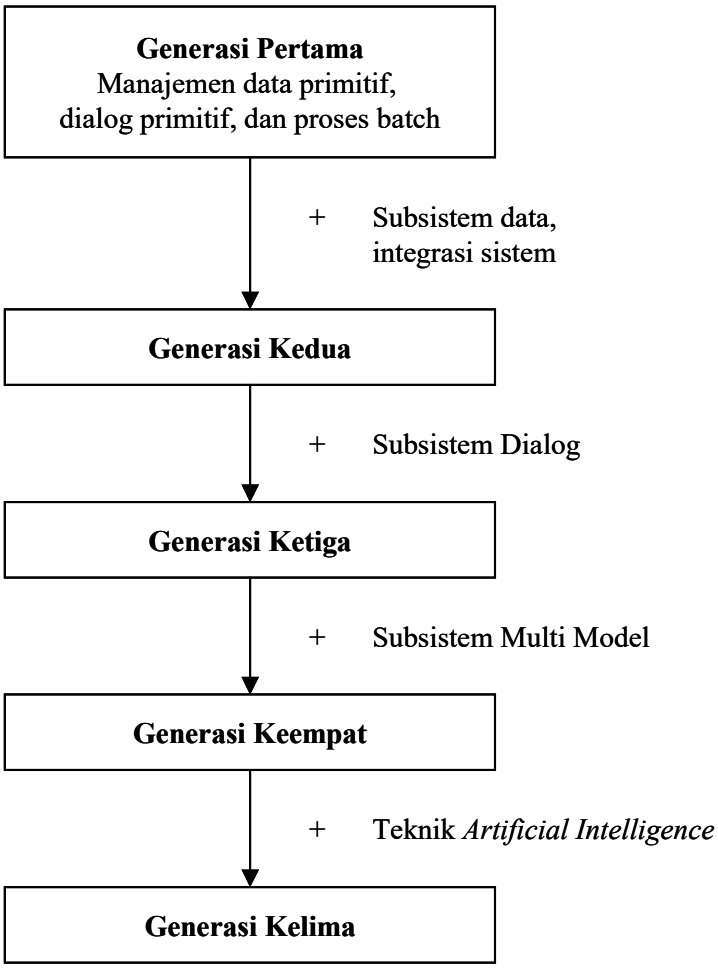
judul "*Models and Managers; The Concept of a Decision Calculus*" mengidentifikasikan kriteria untuk merancang model dalam mendukung manajemen pengambilan keputusan. Kriteria-kriteria tersebut adalah kekuatan, kemudahan pengawasan, kesederhanaan, dan kesempurnaan dari data yang relevan.

Peter G. W. Keen dan Michael S. Scott Morton menulis buku SPK yang berjudul *Decision Support Sistems: An Organizational Perspective* yang dipublikasikan pada tahun 1978. Pada bukunya memberikan penjelasan tentang analisa, perancangan, implementasi, evaluasi dan pengembangan SPK.

Buku *Building Effective Decision Support System* karya Ralph Sprague dan Eric Carlson (1982) menjadi suatu arti penting bagi perkembangan SPK. Buku ini memberikan tuntunan praktis, pemahaman tentang pengorganisasian dalam perancangan SPK. Baonczek, Holsapple, dan Whinston (1981) dalam buku *Foundations of Decision Support System* membahas bidang yang lebih luas pada SPK dan memberi bidang kerja yang nyata untuk merancang SPK. Pada pertengahan dan akhir tahun 1990-an Sistem Informasi Eksekutif dan SPK kelompok dibangun dengan model yang berorientasi pada SPK.

Pada saat ini, sejumlah disiplin ilmu memberikan substansi penemuan untuk pengembangan dan penelitian SPK. Penelitian pada bidang basis data mengkontribusikan perangkat pengolahan data. Ilmu Manajemen mengembangkan model untuk digunakan dalam SPK dan memberikan keunggulan SPK dalam memecahkan masalah. Beberapa perkembangan lain yang berhubung dengan pengembangan SPK, antara lain mencakup intelegensia buatan (*artificial intelegence*), interaksi manusia-komputer, metoda simulasi, perangkat lunak, telekomunikasi.

Sedangkan, perkembangan keilmuan SPKKM dimulai pada awal tahun 1970-an. Pada awalnya SPKKM diimplementasikan dengan cara "primitif". Sistem pertama dikembangkan sesuai dengan keterbatasan tingkat teknologi informasi.



Gambar 1.2 Generasi SPKKM (Siskos dan Spyridakos, 1999)

Lima generasi SPKKM telah dikembangkan dan digunakan pada beberapa SPKKM dengan memanfaatkan keunggulan bidang teknologi informasi dan kemajuan metoda analisis kriteria majemuk. Generasi ketiga dan keempat menghasilkan produktivitas sistem yang tinggi, dan telah berhasil menghasilkan sistem yang menyediakan karakteristik dan fasilitas yang telah digunakan secara efektif oleh pengambil keputusan dan analis keputusan.

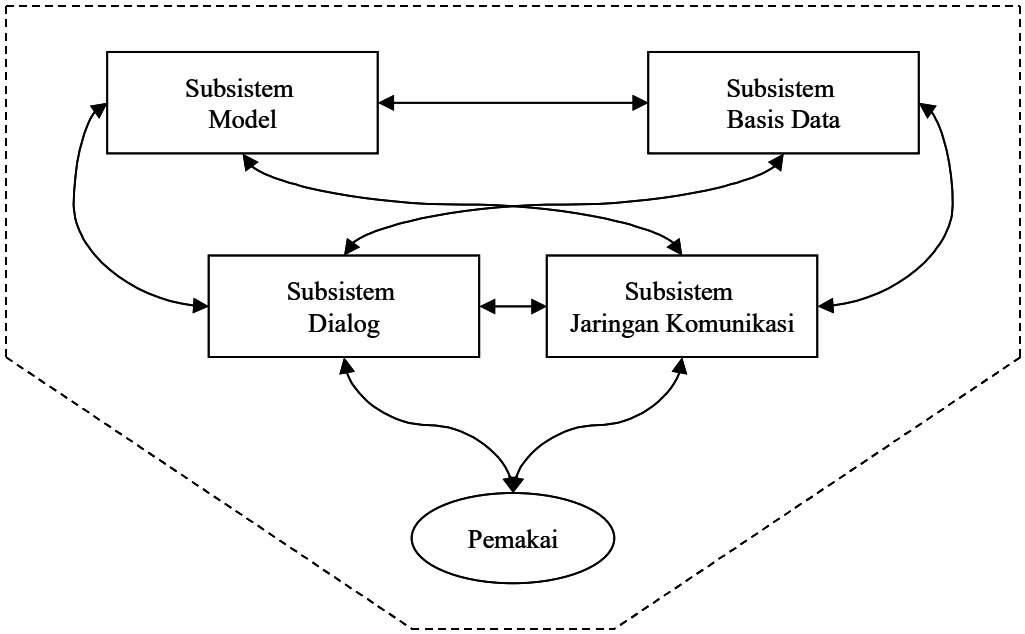
Keunggulan sistem pada generasi ini sesuai dengan kapabilitas metodologi dan perangkat lunak yang ditunjukkan oleh ELECTRE I (Roy dan Skalka, 1985), VIG (Korkohen, 1990), ELECTRE III, IV (Skalka et. al., 1992), PROMETHEE (Brans dan Mareschal. 1989), PREFCALC (Jacquet-Lagrèze, 1984, 1990), MINORA (Siskos dan Yannacopoulos, 1985; Siskos et. al., 1993), LBS (Jaszkiewicz dan Slowinski, 1995), EXPERT CHOICE (Selly dan Forman, 1986), ELECTRE TRI (Yu, 1992), ADELAIS (Siskos dan Despotis, 1989), CAMOS (Osyczka, 1994), dan MIIDAS (Siskos et. al., 1999).

Pada umumnya struktur SPK (sampai pada generasi keempat) dirancang berdasarkan tiga komponen teknologi utama, yaitu (a) Subsistem basis data, (b) subsistem model, dan (c) subsistem dialog. Hal ini, juga ditunjukkan oleh beberapa penelitian yang dituangkan dalam bentuk disertasi, yang pernah dilakukan dalam bidang SPK, seperti Liu (1992), Li (1990), Murillo (1990), atau Suryadi (1992) menunjukkan bahwa perancangan SPK memiliki fokus dalam ketiga komponen tersebut. Hal yang sama terjadi pada perancangan SPKKM, Siskos dan Spyridakos (1999) menyatakan bahwa hampir seluruh SPKKM memiliki kemiripan dalam struktur sistem, yaitu dengan menggunakan struktur sistem berdasarkan tiga komponen teknologi utama, yaitu (a) Subsistem basis data, (b) subsistem model, dan (c) subsistem dialog.

Batasan konsep komponen teknologi SPK dikembangkan berdasarkan pada tingkat teknologi. Pada beberapa tahun belakangan ini, terjadi penambahan batasan teknologi, seperti proses penggalian data, proses analisa *on-line*, *groupware*, dan *knowledgeware* yang telah digunakan pada SPK, dan diperluas untuk mendukung proses pengambilan keputusan (Power, 2000).

Penambahan batasan ini menuntun penelitian untuk mengkategorikan jaringan komunikasi sebagai salah satu komponen teknologi SPK. Penambahan komponen ini dilandasi oleh kebutuhan SPKKM yang dibangun pada penelitian ini sebagai SPK Kelompok (*Group Decision Support System*).

Sesuai dengan karakternya sebagai perangkat pendukung keputusan kelompok, SPK memerlukan data yang memiliki sifat kekinian (*up to date*) yang bisa jadi keberadaannya berada di luar area lokal, atau dengan kata lain, demi efisiensi dan efektivitas dalam proses pengambilan keputusannya, SPK yang dirancang memerlukan perangkat jaringan komunikasi. Sehingga, komponen teknologi SPK yang dikembangkan pada penelitian ini meliputi komponen model, menu dialog, basis data dan jaringan komunikasi.



Gambar I.3 Komponen Teknologi SPKKM pada Penelitian

Sehingga, apabila SPKKM yang dirancang pada penelitian ini dikaitkan dengan tingkat generasi SPKKM, maka SPKKM yang dirancang paling tidak telah melewati ke generasi keempat.

4 Pembatasan Masalah

a Pengembangan model yang dilakukan pada penelitian ini ditujukan untuk permasalahan-permasalahan yang spesifik, masalah yang dapat diselesaikan oleh model adalah permasalahan yang sesuai dengan aksioma dan/atau asumi model.

b Pengembangan model dilakukan berbasis pada kesederhanaan dan kemudahan pengunaan model, yang pada sisi lain mengandung banyak kelemahan.

c Perancangan SPKKM yang dimaksud pada penelitian ini adalah perancangan sistem yang mengintegrasikan subsistem (komponen teknologi) basis data, model, dialog, dan jaringan komunikasi untuk sistem yang terkomputerisasi.

d Proses pengujian model dievaluasi dengan menggunakan studi kasus hipotetik.

5 Tujuan Penelitian

Tujuan penelitian ini adalah pengembangan model AHP, dan perancangan SPKKM untuk pengambil keputusan individu dan kelompok dalam menetapkan nilai bobot kriteria dan penentuan prioritas/ pilihan alternatif.

6 Kegunaan Penelitian

Secara umum, penelitian ini diharapkan mampu memberikan sumbangan terhadap perkembangan ilmu atau menambah khasanah pengetahuan (*body of knowledge*) di bidang teori pengambilan keputusan, sumbangan terhadap pengembangan metodologi perancangan model, metodologi perancangan SPK, dan pemecahan masalah praktis; khususnya pada persoalan-persoalan PKKM.

Pada bidang teori pengambilan keputusan, penelitian ini menghasilkan pendekatan baru dalam hal pengukuran interdependensi kriteria dan analisis pembentukkan konsensus. Dengan demikian, diharapkan produk penelitian dapat digunakan sebagai alternatif model PKKM, serta sumbangan ilmiah yang

mengungkapkan hubungan sebab-akibat pertimbangan kriteria terhadap pemilihan alternatif dalam suatu model pengambilan keputusan.

Kontribusi untuk pengembangan keilmuan pada bidang SPK, penelitian ini mengidentifikasi komponen teknologi tambahan pada SPK, yaitu *jaringan komunikasi*. Sehingga, diharapkan penelitian ini memberikan khasanah baru bagi pengembangan SPK dengan mengintegrasikan komponen jaringan komunikasi ke dalam sistem, disamping komponen model, basis data, dan menu dialog. Kebutuhan ini dilandasi kenyataan bahwa proses pengambilan keputusan tidak jarang dilakukan oleh pengambil keputusan dan/atau responden (pakar) dari jarak jauh.

7 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penyusunan disertasi ini mengikuti alur sebagai berikut:

BAB I: PENDAHULUAN

Bab ini berisi tentang kerangka dasar penyusunan disertasi, yang merupakan pembahasan atas latar belakang masalah, perumusan masalah, posisi keilmuan (*the state of the art*), pembatasan masalah, tujuan penelitian, dan kegunaan penelitian, serta sistematika penulisan disertasi yang dijadikan bagian penutup pada bab ini.

BAB II: TINJAUAN PUSTAKA

Bab ini memuat tentang konsep dasar pengambilan keputusan yang akan menjadi kerangka acuan dalam pengembangan dan rancang bangun SPK, yang menyajikan pembahasan tentang pengambilan keputusan, dan konsep teoritikal tentang Sistem Pendukung Keputusan.

BAB III: METODOLOGI PENELITIAN

Bab ini memuat metodologi pengembangan model dan perancangan SPKKM, yang meliputi sistematika dan tahapan dalam pengembangan model dan perancangan SPKKM.

BAB IV: PENGEMBANGAN MODEL

Pada bab ini disajikan kajian tentang pengembangan model AHP, mengingat pengembangan model ini berbasis AHP, maka sistematika penyajian pada bab ini diawali dengan deskripsi tentang AHP, kemudian dilanjutkan dengan deskripsi pengembangan model.

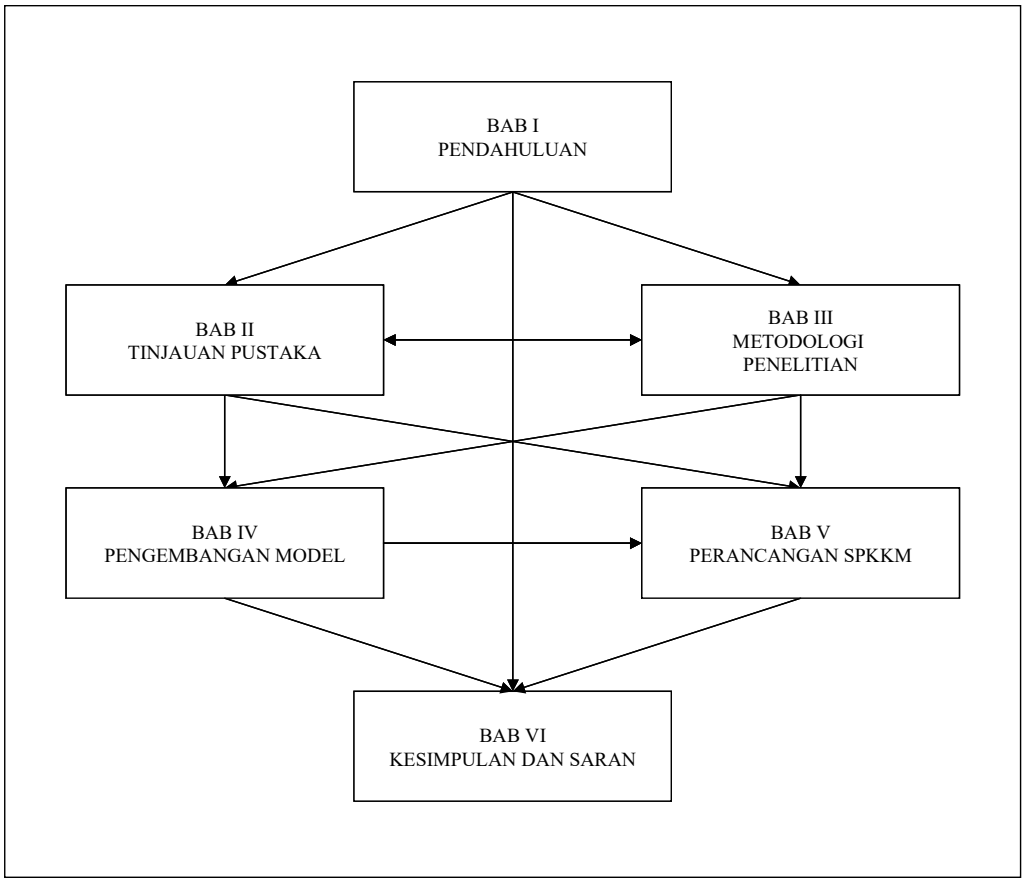
BAB V: PERANCANGAN SISTEM PENDUKUNG KEPUTUSAN KRITERIA MAJEMUK

Dalam bab ini diuraikan pembahasan atas analisis sistem, desain SPKKM yang disertai dengan pendekatan pemrograman komputer.

BAB VI: KESIMPULAN DAN SARAN

Bab ini merupakan bab penutup yang berisi kesimpulan pengembangan model keputusan, dan SPKKM serta saran-saran yang diharapkan digunakan sebagai bahan pertimbangan bagi kegiatan-kegiatan penelitian lebih lanjut (rekomendasi penelitian).

Sistematika penulisan dan keterkaitan antar bab membentuk diagram seperti tampak pada Gambar I.4.



Gambar I.4 Keterkaitan antar Bab dalam Penulisan Disertasi

BAB II
TINJAUAN PUSTAKA

1 Pendekatan Pengambilan Keputusan

Menghadapi segala proses yang terjadi di sekelilingnya dan di dalam dirinya, hampir setiap saat manusia membuat atau mengambil keputusan dan melaksanakannya, ini tentu dilandasi asumsi bahwa segala tindakannya secara sadar merupakan pencerminan hasil proses pengambilan keputusan dalam pikirannya; sehingga sebenarnya manusia sudah sangat terbiasa dalam membuat keputusan. Proses sejak identifikasi masalah sampai pemilihan solusi terbaik inilah yang disebut proses pengambilan keputusan (Putro dan Tjakraatmadja, 1998).

Jika keputusan yang diambil tersebut perlu dipertanggungjawabkan kepada orang lain atau prosesnya memerlukan pengertian pihak lain, maka perlu untuk diungkapkan sasaran yang akan dicapai berikut kronologi proses pengambilan keputusannya (Mangkusubroto dan Tresnadi, 1987).

Proses pengambilan keputusan di dalam kehidupan organisasi adalah suatu proses yang selalu terjadi, dimana hal ini merupakan denyut nadi jalannya organisasi tersebut (Sudirman, 1998). Dari beberapa definisi pengambilan keputusan yang ditemukan, dapat dirangkum bahwa pengambilan keputusan di dalam suatu organisasi merupakan hasil suatu proses komunikasi dan partisipasi yang terus menerus dari keseluruhan organisasi. Hasil keputusan tersebut dapat merupakan pernyataan yang disetujui antar alternatif atau antar prosedur untuk mencapai tujuan tertentu. Pendekatannya dapat dilakukan, baik melalui pendekatan yang bersifat individual/kelompok, sentralisasi/ desentralisasi, partisipasi/tidak berpartisipasi, maupun demokratis/konsensus (Suryadi dan Ramdhani, 1998).

Persoalan pengambilan keputusan, pada dasarnya adalah bentuk pemilihan dari berbagai alternatif tindakan, yang mungkin dipilih, yang prosesnya melalui mekanisme tertentu, dengan harapan akan menghasilkan sebuah keputusan yang terbaik. Penyusunan model keputusan adalah suatu cara untuk mengembangkan hubungan-hubungan logis yang mendasari persoalan keputusan ke dalam suatu model matematis, yang mencerminkan hubungan yang terjadi diantara faktor-faktor yang terlibat.

Keberhasilan seorang pengambil keputusan tergantung dari kemampuannya mengumpulkan fakta dari faktor-faktor krisis, melakukan analisa dan pengamatan atas berbagai perubahan, baik dari dalam maupun luar organisasi (Dutta dan Heda, 2000).

Selain itu, keputusan dapat dilihat pada kaitannya dengan proses, yaitu bahwa suatu keputusan ialah keadaan akhir dari suatu proses yang lebih dinamis, yang diberi label *pengambilan keputusan.* Ia dipandang sebagai proses karena terdiri atas satu seri aktivitas yang berkaitan dan tidak hanya dianggap sebagai tindakan bijaksana. Dengan kata lain, keputusan merupakan sebuah kesimpulan yang dicapai sesudah dilakukan pertimbangan, yang terjadi setelah satu kemungkinan dipilih, sementara yang lain dikesampingkan. Dalam hal ini, yang dimaksud dengan pertimbangan ialah menganalisis beberapa kemungkinan atau alternatif, sesudah itu dipilih satu di antaranya (Salusu, 1996)

Pada akhirnya dapat dikatakan bahwa setiap keputusan itu bertolak dari beberapa kemungkinan atau alternatif untuk dipilih. Setiap alternatif membawa konsekuensi-konsekuensi. Ini berarti, sejumlah alternatif itu berbeda satu dengan yang lain mengingat perbedaan dari konsekuensi-konsekuensi yang akan ditimbulkannya (Simon, 1960). Pilihan yang dijatuhkan pada alternatif itu harus dapat memberikan kepuasan, karena inilah yang merupakan salah satu aspek paling penting dalam keputusan.

Simon (1960) mengajukan model yang menggambarkan proses pengambilan keputusan. Proses ini terdiri dari tiga fase, yaitu:

a *Intelligence*

Tahap ini merupakan proses penelusuran dan pendeteksian dari lingkup problematika serta proses pengenalan masalah. Data masukan diperoleh, diproses, dan diuji dalam rangka mengidentifikasikan masalah.

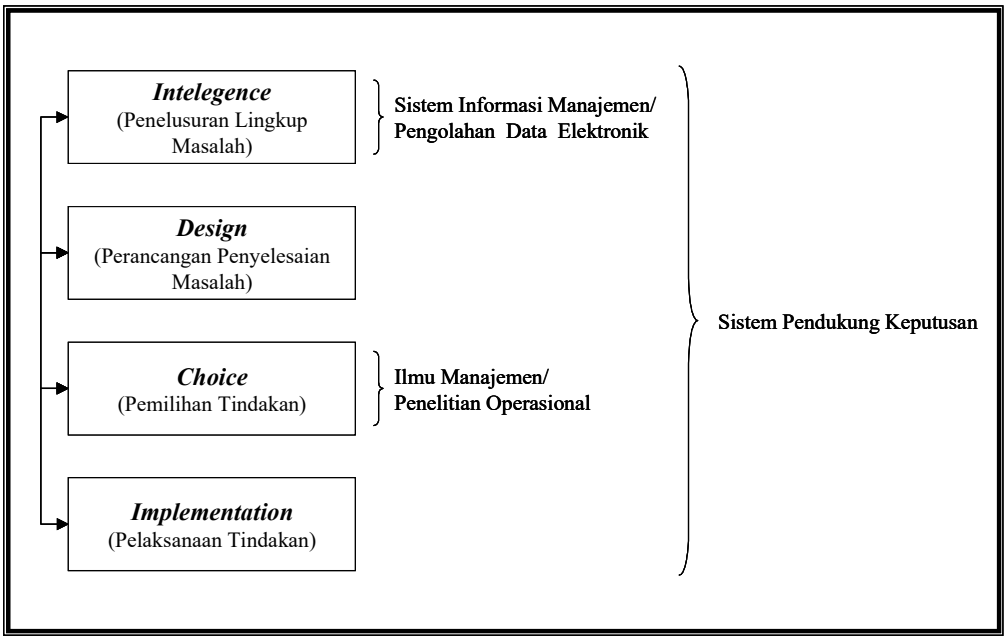
b *Design*

Tahap ini merupakan proses menemukan, mengembangkan dan menganalisis alternatif tindakan yang bisa dilakukan. Tahap ini meliputi proses untuk mengerti masalah, menurunkan solusi dan menguji kelayakan solusi.

c *Choice*

Pada tahap ini dilakukan proses pemilihan diantara berbagai alternatif tindakan yang mungkin dijalankan. Hasil pemilihan tersebut kemudian diimplementasikan dalam proses pengambilan keputusan.

Meskipun implementasi termasuk tahap ketiga, namun ada beberapa pihak berpendapat bahwa tahap ini perlu dipandang sebagai bagian yang terpisah guna menggambarkan hubungan antar fase secara lebih komperhensif. Dalam hal ini, Model Simon juga menggambarkan kontribusi Sistem Informasi Manajemen (SIM) dan Ilmu Manajemen/Penelitian Operasional (IM/PO) terhadap proses pengambilan keputusan, seperti terlihat pada Gambar II.1.



Gambar II.1 Fase Proses Pengambilan Keputusan (Simon, 1960)

Dari deskripsi ketiga tahap pada Gambar II.1, jelas bahwa Pengolahan Data Elektronik (PDE) dan SIM mempunyai kontribusi dalam fase *intelligence*, sedangkan IM/PO berperan penting dalam fase *choice*. Tidak nampak pendukung yang berarti pada tahap *design*, walaupun pada kenyataannya fase ini merupakan salah satu kontribusi dasar dari suatu Sistem Pendukung Keputusan.

Sementara itu, Rosenhead (1989) di dalam Putro dan Tjakraatmadja (1998) menyatakan bahwa tahap-tahap umum yang digunakan pada proses pengambilan keputusan terdiri dari lima tahap pengambilan keputusan, seperti berikut:

a Identifikasi tujuan,
b Identifikasi alternatif pilihan strategi/kegiatan,
c Perkiraan dampak dari setiap kegiatan maupun tujuan,
d Evaluasi konsekuensi, dalam bentuk skala penilaian,
e Pilih alternatif yang memberikan benefit terbesar.

Jauch dan Glueck (1995) dan Salusu (1996) mengemukakan tiga pendekatan utama dalam pengambilan keputusan, yaitu model rasional analitis, intuitif emosional, dan perilaku politis berdasarkan suatu fakta (*fact*) dan nilai (*value*).

1.1 Rasional Analitis

Pengambil keputusan rasional analitis mempertimbangkan semua alternatif dengan segala akibat dari pilihan yang diambilnya, menyusun segala akibat dan memperhatikan skala pilihan (*scale of preferences*) yang pasti, dan memilih alternatif yang memberikan hasil maksimum. Cara ini dikritik, karena pengambil keputusan hanya merupakan bagian dari situasi keputusan yang banyak pihak, tidak cukup memiliki informasi untuk mempertimbangkan semua alternatif dan semua akibatnya, dan tidak memaksimumkan sasaran dalam pikirannya, disamping sasaran dapat berubah.

Pendekatan ini merupakan model klasik dalam pengambilan keputusan bidang ekonomi dan bisnis. Model ini banyak memperoleh kritik karena dianggap kurang realistik karena hanya mempertimbangkan informasi-informasi yang diterima dengan mengabaikan beberapa pertimbangan lainnya.

Pengambilan keputusan yang berdasar logika ialah suatu "studi yang rasional" terhadap semua unsur pada setiap sisi dalam proses pengambilan keputusan. Unsur-unsur itu diperhitungkan secara matang, sambil semua informasi yang masuk dipertimbangkan tingkat kesahihannya. Kemudian, untung rugi dari setiap tindakan yang direncanakan dianalisis secara komprehensif. Pendekatan ini menuntut pengambil keputusan untuk menyingkirkan selera-selera pribadi.

1.2 Intuitif Emosional

Pengambil keputusan intuitif emosional menyukai kebiasaan dan pengalaman, perasaan yang mendalam, pemikiran yang reflektif dan naluri dengan menggunakan proses alam bawah sadar. Proses ini dapat didorong oleh naluri,

orientasi kreatif, dan konfrontasi kreatif. Pengambil keputusan mempertimbangkan sejumlah alternatif dan peluang, secara serempak meloncat dari satu langkah dalam analisis atau mencari yang lain dan kembali lagi. Mereka yang menentang pendekatan ini mengemukakan bahwa cara ini tidak secara efektif menggunakan semua sarana yang ada bagi pengambil keputusan modern.

Model pengambil keputusan yang menggunakan intuisinya seringkali dikritik, karena kurang mengadakan analisis yang terkendali dengan perhatian hanya ditujukan pada beberapa fakta, dengan melupakan banyak elemen penting. Dalam pengambilan keputusan dengan menggunakan intuisi tidak banyak tergantung pada fakta yang lengkap. Mungkin dengan informasi yang sedikit saja seseorang sudah dapat mengambil keputusan karena intuisi itulah yang dominan.

1.3 Perilaku Politis

Berbeda dengan model-model pendekatan yang telah diuraikan sebelumnya, bahwa cara pengambilan keputusan perilaku politis merupakan pengambilan keputusan individual dengan melakukan pendekatan kolektif. Juga dianggap teori deskriptif yang menyarankan agar organisasi tempat pengambil keputusan bekerja membatasi pilihan yang ada. Keputusan diambil kalau beberapa orang yang terlibat dalam proses itu menyetujui bahwa mereka telah menemukan pemecahan. Mereka melakukan hal ini dengan saling menyesuaikan diri dan saling berunding mengikuti peraturan permainan cara mengambil keputusan dalam organisasi pada masa lalu. Pengambil keputusan harus mempertimbangkan apakah hasil keputusan itu dapat dilaksanakan secara politis.

2 Pengambilan Keputusan Individu dan Kelompok

Pada dasarnya pengambilan keputusan kelompok berdasar pada pengambilan keputusan secara individu anggota kelompok. Berikut ini disajikan model-

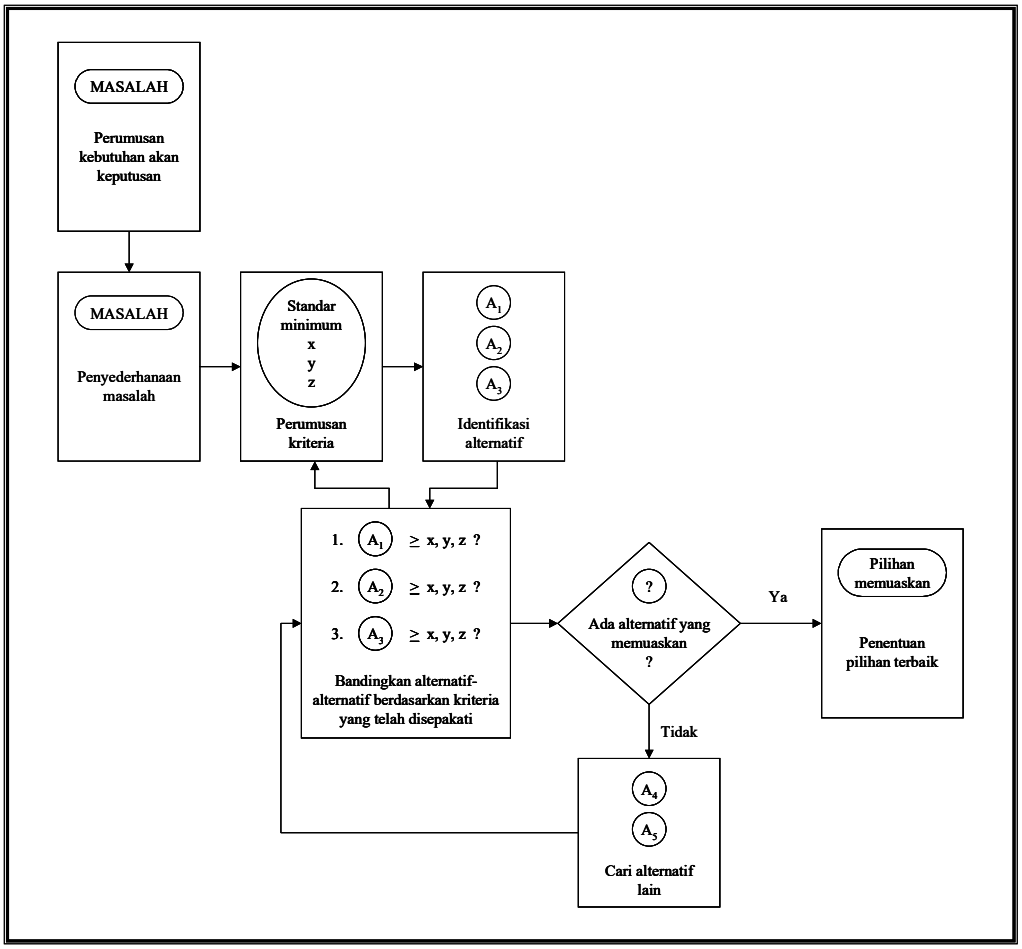
model pengambilan keputusan individual yang dikemukakan oleh Robbins (1991), dengan pendekatan *contingency* (model pengambilan keputusan yang dipilih dan digunakan sesuai dengan situasi tertentu), antara lain sebagai berikut.

2.1 Pengambilan Keputusan Individual

2.1.1 The Satisfying Model

Esensi dari *the Satisfying model*, pada saat dihadapkan pada masalah kompleks, pengambil keputusan berusaha menyederhanakan masalah-masalah pelik sampai pada tingkat di mana dia siap untuk memahaminya. Hal ini dikarenakan secara manusiawi dia tidak mungkin memahami dan mencerna semua informasi penting secara optimal. Di dalam model ini pembatasan proses pemikiran dirahkan pada pengambilan keputusan dengan rasionalitas terbatas (*bounded rationality*), yaitu proses penyederhanaan model dengan mengambil inti masalah yang paling esensial tanpa melibatkan seluruh permasalahan yang konkrit.

Rasionalitas terbatas adalah batas-batas pemikiran yang memaksa orang membatasi pandangan mereka atas masalah dan situasi. Pemikiran itu terbatas karena pikiran manusia tidak memiliki kemampuan untuk memisahkan dan mengolah informasi yang bertumpuk. Bagi para pengambil keputusan daripada mempertimbangkan enam atau delapan alternatif, lebih baik cukup bekerja dengan dua atau tiga alternatif untuk mencegah kekacauan. Pada dasarnya, manusia sudah berpikir logis dan rasional, tetapi dalam batas-batas yang sempit.



Gambar II.2 *The Satisfying Model* (Robbins, 1991).

Faktor-faktor yang menyebabkan timbulnya rasionalitas terbatas, antara lain informasi yang datang dari luar sering sangat kompetitif atau informasi itu tidak sempurna, kendala waktu dan biaya, serta keterbatasan seorang pengambil keputusan yang rasional untuk mengerti dan memahami masalah dan informasi. Konsep ini memberi tekanan pada batas-batas dari rasionalitas pengambilan keputusan, di samping dapat menjelaskan mengapa dua orang yang menggunakan informasi sama, bisa menghasilkan keputusan yang berbeda.

Langkah-langkah model pengambilan keputusan ini adalah sebagai berikut:

1 Penetapan tujuan (kebutuhan) pengambilan keputusan berkaitan dengan adanya masalah tertentu.
2 Menyederhanakan masalah.
3 Penetapan standar minimum dari serangkaian kriteria keputusan.
4 Mengidentifikasi serangkaian alternatif yang dibatasi.
5 Menganalisis dan membandingkan setiap alternatif, apakah memenuhi kendala lebih besar atau sama (≥) dengan standar minimum dari serangkaian keputusan.
6 Apakah alternatif yang memenuhi syarat keputusan itu ada?
7 Jika ya, dipilih salah satu alternatif yang dianggap terbaik.
8 Jika tidak, dilakukan kembali pencarian alternatif seperti pada langkah ke-5.

2.1.2 The Optimizing Decision Making Model

Dalam model ini, seorang pengambil keputusan yang penuh keyakinan berusaha menyusun alternatif-alternatif, memperhitungkan untung rugi dari setiap alternatif itu terhadap tujuan organisasi. Sesudah itu ia memperkirakan kemungkinan timbulnya bermacam-macam kejadian di kemudian hari, mempertimbangkan dampak dari kejadian-kejadian itu terhadap alternatif-alternatif yang telah dirumuskan, dan kemudian menyusun urut-urutannya secara sistematis sesuai prioritas. Barulah ia membuat keputusan. Keputusan yang dibuatnya itu dianggap optimal karena setidaknya ia telah memperhitungkan semua faktor yang berkaitan dengan keputusan tersebut.

Model ini menggambarkan bagaimana individu harus memaksimalkan hasil dari keputusan yang diambilnya. Lima tahap/langkah yang harus diikuti, baik secara implisit maupun eksplisit dalam proses keputusan menurut model ini, yaitu:

a Tegaskan kebutuhan untuk suatu keputusan,
b Identifikasi kriteria keputusan,

c Alokasikan bobot nilai pada kriteria,

d Kembangkan berbagai alternatif,

e Evaluasi alternatif-alternatif tersebut di atas,

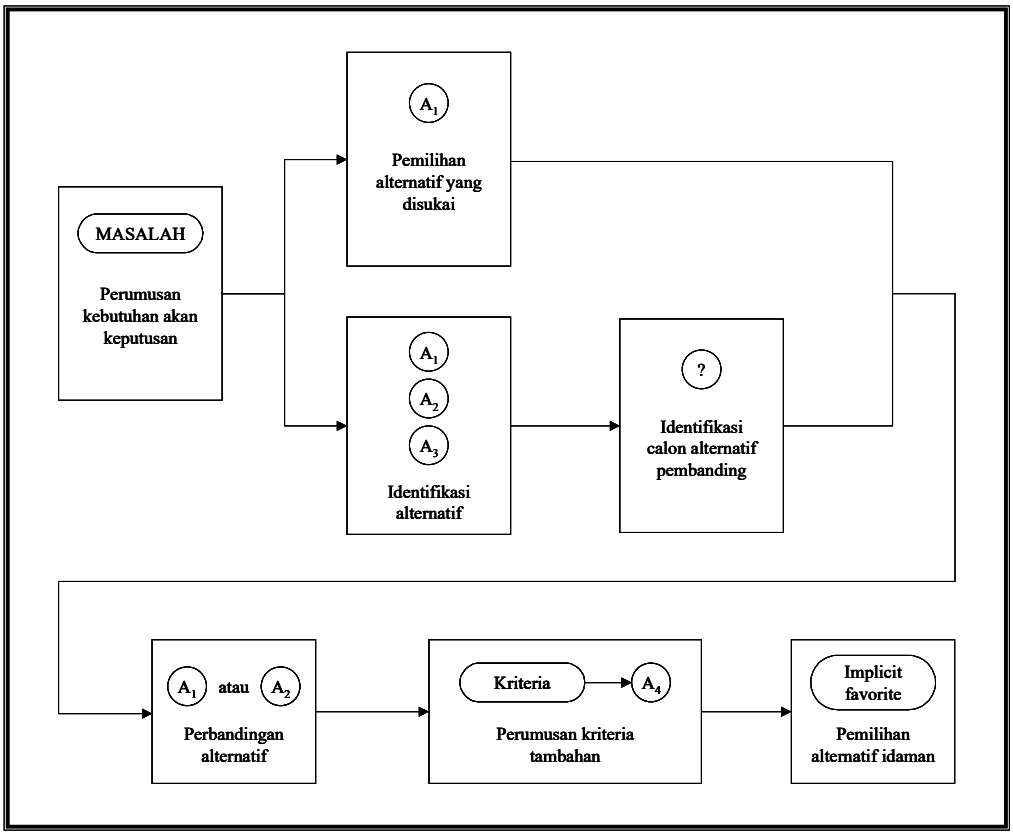
f Pilih alternatif terbaik.

Asumsi untuk *the optimizing model*, yaitu:

- Berorientasi pada tujuan,
- Pengambil keputusan dapat mengenal semua kriteria yang relevan dan dapat menyusun daftar dari semua alternatif yang masih aktif dan nyata,
- Secara rasional semua kriteria dan alternatif disesuaikan dengan tujuannya,
- Pengambil keputusan yang rasional akan memilih peringkat tertinggi yang akan memberikan manfaat maksimum.

2.1.3 The Implicit Favorite Model

Favorite model dirancang dalam kaitan dengan keputusan kompleks dan tidak rutin. Seperti halnya pada model satisfying, pada model ini pun menyangkut proses penyederhanaan masalah yang kompleks oleh individu pembuat keputusan. Bedanya dengan satisfying model, bahwa *the implicit favorite model* tidak memasuki tahap pengambilan keputusan melalui pengevaluasian alternatif yang cukup sulit karena perlu rasional dan obyektif. Pada awal proses keputusan, si pengambil keputusan sudah cenderung memilih alternatif yang dirasakan paling baik/disukai.



Gambar II.3 *The Implicit Favorite Model* (Robbins, 1991).

Langkah-langkahnya adalah sebagai berikut:

a Penentuan kebutuhan untuk pengambilan keputusan karena ada masalah.

b Mengidentifikasi alternatif dan langsung menetapkan pilihan satu alternatif menurut preferensinya.

c Mengidentifikasi alternatif lain, kemudian dipilih lagi satu alternatif lain sebagai pembanding untuk mengukuhkan alternatif favorit.

d Pemilihan alternatif yang menjadi idaman si pengambil keputusan.

2.2 Pengambilan Keputusan Kelompok

Keperluan pengambilan keputusan kelompok terjadi manakala pemilik keputusan (pengambil keputusan) secara subjektif dilibatkan sebagai partisipan dalam pengambilan keputusan, yang pada beberapa kasus dia tidak telibat atau mengetahui segala aktivitas secara utuh (Brugha, 1998). Untuk dapat menggambarkan permasalahan secara utuh, dalam evaluasi suatu keputusan

maka seorang pengambil keputusan perlu mempertimbangkan pendapat orang lain yang memahami suatu permasalahan.

Proses pengambilan keputusan kelompok pada umumnya menghasilkan keputusan yang kompleks. Masing-masing pihak yang terlibat proses pengambilan keputusan, bisa memiliki nilai-nilai dan prinsip-prinsip yang berbeda. Perbedaan nilai atau prinsip hidup ini, bisa menimbulkan perbedaan/pertentangan tujuan/kepentingan, sehingga dapat menimbulkan perbedaan preferensi di antara pihak yang terlibat, situasi tersebut disebut konflik. Di dalam situasi konflik, idealnya satu pihak tidak memaksakan keinginan (strategi) untuk mencapai keinginannya sendiri, melainkan harus mau dan mampu bekerja sama (*win-win*) melalui negosiasi. Proses kerjasama yang dilandasi oleh niat yang tulus dan terbuka, akan mencapai hasil yang saling menguntungkan, bahkan bisa memberi efek sinergi, sehingga hasil yang akan didapat oleh masing-masing pihak akan lebih baik daripada proses yang bersifat kompetitif (*win-loose*) (Putro dan Tjakraatmadja, 1998)

Beberapa metode pengambilan keputusan kelompok antara lain adalah sebagai berikut:

2.2.1 Nash Bargaining Solution

Salah satu cara memandang masalah keputusan kelompok adalah tawar menawar (*bargaining*). Nash merumuskan masalah tawar menawar ini sampai kepada solusinya. Hasilnya adalah para pelaku harus meningkatkan produk yang bermanfaat bagi mereka masing-masing (*product individual utilities*). Peranan solusi Nash tersebut adalah menghitung sejauhmana keuntungan relatif dari suatu tawar menawar dengan nilai dasar yang akan berlaku, bila tidak ada kesepakatan. Pendekatan Nash didasarkan pada pengertian bersaing dari pembuat keputusan kelompok dan solusi keseimbangan (*equilibrium*) terhadap masalah tawar menawar. Dampak ancaman dari masing-masing

pelaku ikut dipertimbangkan. Masing-masing individu mencari kebaikan untuk kepentingan diri sendiri dan atau kelompoknya (Robbins, 1991).

2.2.2 Additive Utility

Pengambilan keputusan ini didasarkan pada langkah lebih baik mencapai kebaikan bersama (kolektif) daripada untuk kebaikan individual yang tidak adil yang tidak mencapai tujuan bersama yang diharapkan. Fungsi utilitas kelompok merupakan jumlah yang ditimbang dari utilitas individual adalah:

$$U_{(a)} = u_1 (a_1) + u_2 (a_2) + . \ . \ . + u_n (a_n) \qquad \ldots(II.1.)$$

dimana $a = (a_1, a_2, ..., a_n)$ adalah vektor alternatif, dan u_i adalah bobot utilitas elemen ke-i.

Asumsi tentang peraturan keputusan kelompok adalah:

a Preferensi sosial (kelompok) memenuhi ketetapan untuk memaksimalkan utilitas yang diharapkan

b Preferensi individual memenuhi ketetapan untuk memaksimalkan utilitas yang diharapkan.

c Bila dua buah prospek P dan Q sama baiknya dari sudut pandang setiap individu, hal ini juga sama baiknya dari sudut pandang sosial (kelompok).

2.2.3 Metoda Delphi

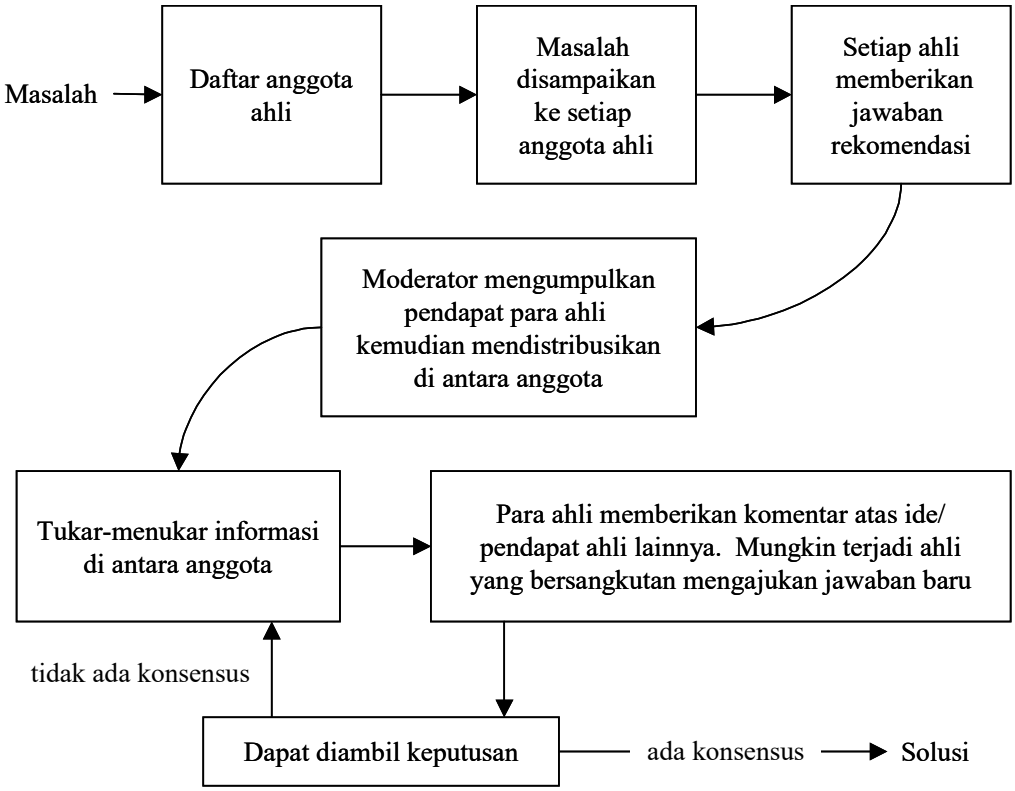
Menurut Halim et. al. (1996) metoda Delphi adalah suatu teknik membuat keputusan yang dibuat oleh suatu kelompok, dimana anggotanya terdiri dari para ahli atas masalah yang akan diputuskan. Metoda ini secara ekstensif dikembangkan oleh Olaf Helmer dan Rand Corporation.

Proses penetapan keputusan menggunakan metoda Delphi dimulai dengan identifikasi permasalahan yang akan dicari penyelesaiannya. Permasalahan ini

kemudian dijelaskan kepada para ahli yang akan dilibatkan dalam proses Delphi. Para ahli yang terlibat pada proses pembuatan keputusan ini berkomunikasi satu dengan yang lainnya, diatur oleh seorang "moderator".

Setelah masing-masing ahli mempelajari masalah yang dibahas, kemudian masing-masing memberikan jawaban/rekomendasi secara independen sesuai dengan pendapatnya; dikumpulkan oleh moderator. Selanjutnya moderator mendistribusikan setiap pendapat dari seorang ahli kepada anggota ahli lainnya, sehingga semua anggota dapat mempelajari pendapat anggota lainnya. Pada tahap ini terjadi proses tukar-menukar informasi (*sharing*) antar anggota, tanpa ada tekanan "diantaranya". Atas dasar proses *sharing* ini, moderator meminta pendapat baru dari seluruh anggota anggota ahli, barangkali mereka akan merubah pendapatnya. Moderator kembali mengumpulkan pendapat pada interasi ke-dua ini.

Pada tahap kedua ini, mungkin ditetapkan suatu jawaban, seandainya ternyata semua anggota sepakat pada satu keputusan. Namun, seandainya di antara anggota ini tidak terjadi kesepakatan, moderator mengulang proses tukar-menukar informasi ini sampai ditemukan kesepakatan.



Gambar II.4 Proses Metoda Delphi

3 Pengambilan Keputusan Kriteria Majemuk

PKKM pada prinsipnya adalah model pengambilan keputusan untuk penentuan prioritas alternatif dengan menggunakan dua atau lebih kriteria/atribut, yang satu sama lain terkadang memiliki konflik, dan kriteria yang tidak sepadan untuk beberapa kepentingan kelompok (Ma et. al., 1999; Malczewski dan Jakson, 2000; dan Eum et. al., 2001).

Untuk menggambarkan paradigma antara pengambilan keputusan kriteria tunggal dan PKKM, berikut ini penggambaran paradigma-paradigma tersebut dalam model matematis.

Paradigma Kriteria Tunggal

$$Z = Max \{ C(a) \mid a \in A \}; A \{ a_1, a_2, ..., a_n \} \quad \ldots(II.2)$$

Paradigma Multikriteria

$$Z = Max \{ c_1(a), c_2(a), c_3(a), ..., c_j(a), ..., c_k(a) \mid a \in A \} \quad \ldots(II.3)$$

dimana: Z = fungsi tujuan

$ci\,(a)$ = nilai preferensi alternatif dengan mempertimbangkan kriteria ke-i

Perbandingan dari setiap alternatif tergantung dari evaluasi alternatif untuk setiap kriteria (Sarkis, 2000). PKKM dikategorikan sebagai permasalahan pengambilan keputusan semi terstuktur (Siskos, 1999).

Kriteria menunjukkan definisi masalah dalam bentuk yang konkrit dan kadang-kadang dianggap sebagai sasaran yang akan dicapai. Analisis atas kriteria penilaian dilakukan untuk memperoleh seperangkat standar pengukuran, untuk kemudian dijadikan sebagai alat dalam memperbandingkan berbagai alternatif yang masuk dalam pertimbangan.

Penggunaan model PKKM untuk suatu keputusan tertentu tergantung pada saat pemilihan kriteria yang digunakan sebagai satuan analisis (Al-Shemmeri et. al., 1997). Pada saat pembuatan kriteria, pengambil keputusan harus mencoba untuk menggambarkan dalam bentuk kuantifikasi jika hal ini memungkinkan. Karena akan selalu ada beberapa faktor yang tidak dapat dikuantifikasikan yang juga tidak dapat diabaikan, maka hal ini mengakibatkan semakin sulitnya membuat perbandingan. Kenyataan bahwa kriteria yang tidak bisa dikuantifikasikan itu sukar untuk diperkirakan dan diperbandingkan hendaknya tidak menyebabkan pengambil keputusan untuk tidak menggunakan kriteria tersebut, karena kriteria ini dapat saja relevan dengan masalah utama di dalam setiap analisis. Beberapa kriteria yang kemungkinan sangat penting, tetapi sulit dikuantifikasi, adalah seperti faktor-faktor sosial (seperti gangguan lingkungan), estetika, keadilan, faktor-faktor politis, serta kelayakan

pelaksanaan. Akan tetapi jika suatu kriteria dapat dikuantifikasi tanpa merubah pengertiannya, maka hal ini harus dilakukan.

Sifat-sifat yang harus diperhatikan dalam memilih kriteria pada setiap persoalan pengambilan keputusan adalah sebagai berikut:

a **Lengkap**, sehingga dapat mencakup seluruh aspek penting dalam persoalan tersebut. Suatu set kriteria disebut lengkap apabila set ini dapat menunjukkan seberapa jauh seluruh tujuan dapat dicapai.

b **Operasional**, sehingga dapat digunakan dalam analisis. Sifat operasional ini mencakup beberapa pengertian, antara lain adalah bahwa set kriteria ini harus mempunyai arti bagi pengambil keputusan, sehingga ia dapat benar-benar menghayati implikasinya terhadap alternatif yang ada. Selain itu, jika tujuan pengambilan keputusan ini harus dapat digunakan sebagai sarana untuk meyakinkan pihak lain, maka set kriteria ini harus dapat digunakan sebagai sarana untuk memberikan penjelasan atau untuk berkomunikasi. Operasional ini juga mencakup sifat dapat diukur. Pada dasarnya sifat dapat diukur ini adalah untuk:

   (1) memperoleh distribusi kemungkinan dari tingkat pencapaian kriteria yang mungkin diperoleh (untuk keputusan dalam ketidakpastian).

   (2) mengungkapkan preferensi pengambil keputusan atas pencapaian kriteria.

c **Tidak berlebihan**, sehingga menghindarkan perhitungan berulang. Dalam menentukan set kriteria, jangan sampai terdapat kriteria yang pada dasarnya mengandung pengertian yang sama.

d **Minimum**, agar lebih mengkomperhensikan persoalan. Dalam menentukan set kriteria perlu sedapat mungkin mengusahakan agar jumlah kriterianya sesedikit mungkin. Karena makin banyak kriteria maka makin sukar pula untuk dapat menghayati dengan baik, jumlah perhitungan yang diperlukan dalam analisis akan meningkat dengan cepat.

Pendekatan umum dalam pemecahan persoalan PKKM adalah bentuk/struktur penggabungan fungsi nilai (Pudinovski, 1999), atau dengan kata lain, preferensi pengambilan keputusan pada dasarnya merupakan gabungan dari preferensi alternatif untuk masing-masing kriteria. Jadi bila c_i (a_j) menyatakan fungsi nilai satu dimensi yang menggambarkan struktur preferensi pengambil keputusan untuk suatu kriteria c_i khusus untuk suatu alternatif a_j, maka preferensi pengambil keputusan atas seluruh kriteria dapat dituliskan sebagai berikut:

$$f(a_j) = \sum_{i=1}^{n} c_i(a_j) \qquad \ldots(II.4)$$

dimana: n = jumlah alternatif

Untuk menghadapi PKKM maka konsep dasar pemilihan dapat didekati melalui konsep dominasi elemen, dengan pernyataan bila penilaian kriteria dituliskan sebagai ($c_1, c_2, c_3, \ldots, c_n$), maka alternatif a_1 disebut mendominasi alternatif a_2 jika dan hanya jika:

$$\sum_{i=1}^{n} c_i(a_1) > \sum_{i=1}^{n} c_i(a_2) \qquad \ldots(II.5)$$

Untuk menghadapi PKKM konsep dasar pemilihan dapat diuraikan sebagai berikut:

a Dominasi

Bila kriteria penilaian dituliskan sebagai (c_1, c_2, ..., c_n), maka alternatif a_1 disebut mendominasi alternatif a_2 jika dan hanya jika:

$$c_i\ (a_1) \geq c_i\ (a_2) \qquad \ldots(II.6)$$

untuk setiap i = 1, 2, 3, ..., n , dan paling sedikit satu $c_i\ (a_1) > c_i\ (a_2)$

Jika terdapat satu alternatif yang mendominasi alternatif yang lain, maka dengan mudah dipilih alternatif terbaik. Tetapi keadaan ini jarang ditemui dalam dunia nyata, yang paling sering terjadi adalah bahwa satu alternatif memiliki nilai yang lebih baik untuk beberapa kriteria, tetapi lebih buruk pada beberapa kriteria yang lainnya. Untuk mengatasi hal ini, terdapat beberapa pendekatan, diantaranya seperti yang diuraikan berikut ini.

b Leksikografi

Cara pemilihan dengan cara ini adalah sebagai berikut:
Alternatif a_1 akan lebih disukai daripada alternatif a_2, jika:

1) $c_i\ (a_1) > c_i\ (a_2)$, atau
2) $c_i\ (a_1) = c_i\ (a_2)$, i = 1, 2, 3, ..., k; dan
   $c_{k+1}\ (a_1) > c_{k+1}\ (a_2)$ …(II.7)

untuk beberapa k = 1, 2, 3, ..., n-1.

Dengan kata lain, alternatif a_1 lebih disukai dari a_2, semata-mata karena untuk kriteria pertama (c_1), alternatif a_1 mempunyai nilai yang lebih baik dari alternatif a_2, tanpa memperhatikan bagaimana baik buruknya nilai pada kriteria yang lain. Baru apabila alternatif a_1 dan a_2 sama baiknya, maka c_2 digunakan sebagai pembanding; dan seterusnya.

Kekurangan metode ini adalah bahwa bila satu alternatif telah mempunyai nilai terbaik pada c_1, maka alternatif tersebutlah yang dipilih, tanpa memperhatikan bagaimana nilai alternatif tersebut pada kriteria lainnya, padahal mungkin amat jelek.

c Tingkat Aspirasi.

Untuk melakukan pemilihan diantara beberapa alternatif, dapat pula ditentukan tingkat aspirasi yang harus dicapai oleh alternatif tersebut. Akan tetapi pada situasi lain, mungkin akan diperoleh bahwa tidak ada satu alternatif pun yang dapat memenuhi seluruh tingkat aspirasi yang ditentukan. Atau sebaliknya, setelah seleksi masih terdapat beberapa alternatif yang memenuhi, sehingga cara ini tidak menjamin diperolehnya suatu alternatif terbaik.

Berdasarkan metoda pendekatannya, PKKM dapat diklasifikasikan pada tiga pendekatan utama (Belton dan Hodgkin, 1999), yaitu:

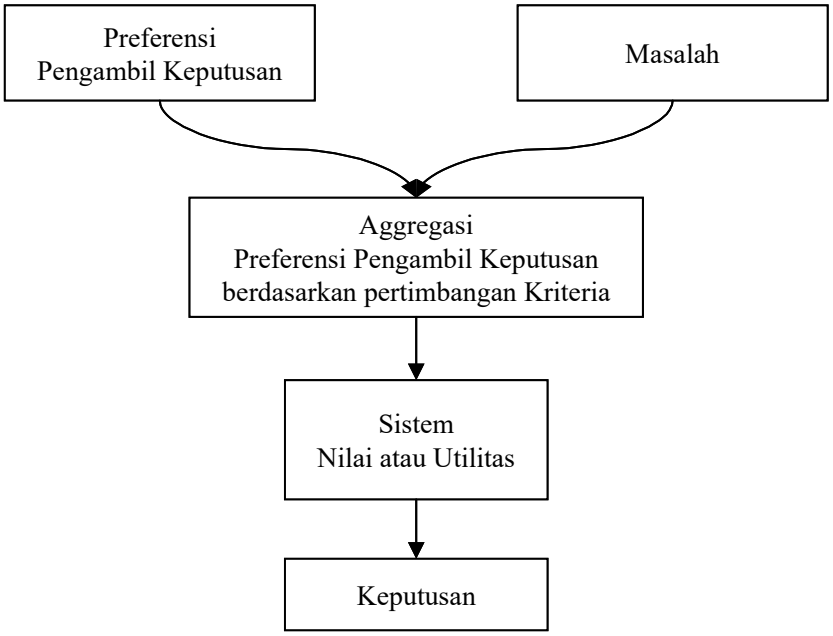
1 Pendekatan MAUT (*Multiple Attribute Utility Theory*), teori ini dibangun atas aksioma dasar bahwa banyak keputusan yang secara tidak sadar atau sadar berusaha memaksimumkan beberapa fungsi. Penggabungan perbedaan pendapat dilibatkan dalam analisis. Dengan kata lain, jika pengambil keputusan ditanya tentang kecenderungannya, maka jawabannya koheren dengan fungsi utilitas (***U***) yang tidak dapat diketahui secara pasti (Vincke et. al., 1992). Bentuk analisa sederhana yang umumnya digunakan dalam MAUT adalah bentuk aditif:

$$U(a) = \sum_{j=1}^{n} U_j(c_j(a)) \qquad \ldots(\text{II.8})$$

dimana : $U(a)$ = fungsi utilitas untuk alternatif a

U_j = peningkatan fungsi nyata (umumnya menggunakan bentuk kriteria dengan skala ukuran yang sama)

c_j = kriteria ke-j



Gambar II.5 Pendekatan Multiple Attribute Utility (Siskos dan Spyridakos, 1999)

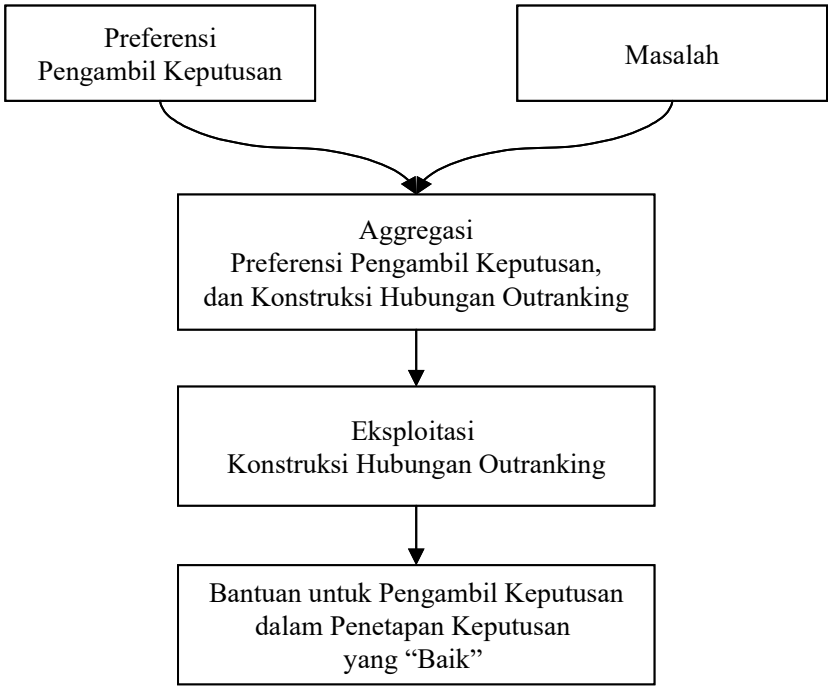
Model-model yang dibangun berdasarkan pendekatan ini misalnya:

a AHP (*Analityc Hierarchy Process*) oleh Thomas L. Saaty (1971);

b UTA (*Utilité Additive*) oleh Jacquet-Lagréze dan Siskos (1982);

c ANP (*Analityc Network Process*) oleh Thomas L. Saaty (1999).

2 *Pendekatan Metoda Outranking*, ide utama pengembangan metoda ini adalah bahwa diperlukan metoda yang menghasilkan solusi yang lebih baik pada model PKKM, dengan penghindaran dari bentuk hipotesa matematik yang terlalu kuat dan pertanyaan yang terlalu sulit bagi pengambil keputusan. Dengan kata lain, metoda outranking dirancang untuk memperluas hubungan dominasi oleh beberapa elemen yang dibiarkan tanpa diskusi, dan membangun kekuatan stabilitas preferensi.

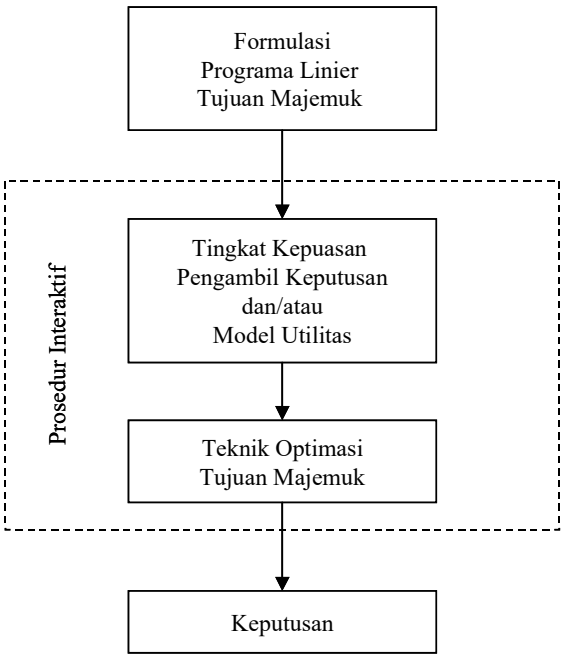


Gambar II.6 Pendekatan Hubungan Outranking (Siskos dan Spyridakos, 1999)

Konsep outranking dirancang pertama kali oleh B. Roy. Roy (1974) mendefinisikan hubungan outranking sebagai hubungan biner antar alternatif, hubungan ini diberikan jika diketahui preferensi pengambil keputusan dan/atau jika diberikan penaksiran kualitas dari alternatif berdasarkan data dari keadaan yang sesungguhnya terjadi, dimana terdapat cukup argumentasi untuk memutuskan bahwa suatu alternatif dinyatakan lebih baik dibanding alternatif lainnya dan tidak ada alasan yang esensial untuk menolak pernyataan ini. Model-model yang dirancang atas dasar pendekatan ini misalnya:

a Metoda Electre I (Roy, 1968)
b Metoda Electre II (Roy dan Bertier, 1971)
c Metoda Electre III (Roy, 1978)
d Metoda Electre IV (Huggonnard dan Roy, 1982)
e Metoda Promethee (Brans dan Vincle, 1985)

3 *Metoda Programa Linier Tujuan Majemuk (Multiobjective Linear Programming)*, metoda ini merupakan perkembangan lebih lanjut dari programa matematis (OR), tujuannya adalah untuk pemecahan masalah dengan nilai alternatif yang tidak ditetapkan, dengan fungsi tujuan lebih dari satu. Solusi dari model ini berupa nilai estimasi dengan prosedur iteratif yang menunjukkan hasil: (a) tingkat kepuasan pengambil keputusan pada kriteria; atau (b) membuat konstruksi model utilitas dari pengambil keputusan, yang digunakan untuk membuat/memilih keputusan berdasarkan prosedur maksimasi utilitas; atau (c) kombinasi dua hasil di atas (Siskos dan Spyridakos, 1999).



Gambar II.7 Pendekatan Optimasi *Programa Linier Tujuan Majemuk* (Siskos dan Spyridakos, 1999)

Pemilihan ketepatan metodologi analisa keputusan pada PKKM tergantung dari banyak faktor, antara lain, sifat masalah, karakteristik pengambil keputusan, ketersediaan perangkat lunak, dan kesederhanaan/kemudahan penggunaan metodologi (Stewart, 1999).

Wierzbicki dan Granat (1999) menyatakan pada proses analisa keputusan untuk PKKM, model secara khusus harus mampu menyajikan dan menganalisa preferensi dari pengambil keputusan yang dinyatakan dalam model preferensi (*prefential model*). Pada sisi lain, menyatakan Easley et. al. (2000) dalam mengevaluasi efektifitas suatu model keputusan kriteria majemuk adalah penting untuk menggunakan kelompok sebagai dasar pengambilan keputusan, dan melakukan evaluasi gambaran preferensi setiap anggota kelompok dalam menilai setiap alternatif.

Lebih lanjut Wierzbicki dan Granat (1999), menyatakan tahapan pengambilan keputusan pada model PKKM adalah sebagai berikut:

a pengenalan dan formulasi masalah, pengumpulan data, dan pemilihan model substantif.
b formulasi model substantif (termasuk validasi model).
c analisis detail dari model substantif
d pemilihan alternatif
e implementasi

Siskos el. al. (1999) menyatakan bahwa metoda PKKM pada prinsipnya harus mampu memberikan solusi terhadap persoalan PKKM untuk satu atau lebih dari pernyataan permasalahan:

a pemilihan alternatif terbaik;
b pembuatan klasifikasi alternatif;
c urutan (*rank*) alternatif dari yang terbaik sampai dengan yang terburuk;
d menggambarkan persoalan.

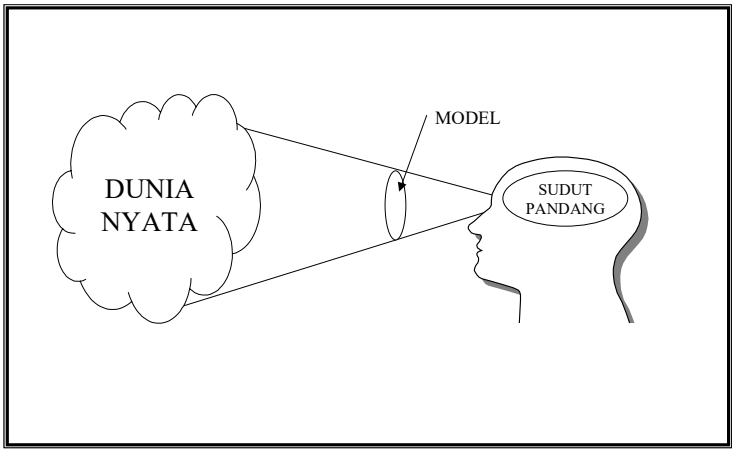
4 Pemodelan

Daerah penciptaan dan penanganan model mempunyai kemampuan kontribusi terbesar pada SPK. Selain itu diperlukan kemampuan analitis yang dilengkapi

oleh sistem dari analisis matematik, statistik, finansial, dan sebagainya (Suryadi dan Ramdhani, 1998).

Untuk keperluan analisa, biasanya sistem digambarkan ke dalam suatu model. Istilah model diartikan sebagai tiruan dari kondisi sebenarnya, atau dengan kata lain, model didefinisikan sebagai representasi atau formalisasi dalam bahasa tertentu (yang disepakati berdasarkan sudut pandang tertentu) dari suatu sistem nyata, atau penyederhanaan dari gambaran sistem yang nyata. Adapun sistem nyata adalah sistem yang sedang berlangsung dalam kehidupan, sistem yang dijadikan titik perhatian dan dipermasalahkan.

Dengan demikian, pemodelan adalah proses membangun atau membentuk sebuah model, dalam bahasa formal tertentu, dari suatu sistem nyata berdasarkan sudut pandang tertentu.



Gambar II.8 Skema Proses Pemodelan

Sistem nyata akan dilihat dan dibaca oleh pemodel dan membentuk citra atau gambaran tertentu di dalam pikirannya. Namun citra ini tidak persis sama dengan sistem nyata, karena pemodel membacanya dengan menggunakan sudut pandang tertentu. Sudut pandang yang dimaksud di sini adalah visi atau wawasan yang dipengaruhi oleh tiga faktor, yakni:

* Sistem nilai yang diyakini/dianut oleh pemodel
* Ilmu pengetahuan yang dimiliki oleh pemodel
* Pengalaman hidup dari pemodel.

Secara umum model digunakan untuk memberikan gambaran (*description*), memberikan penjelasan (*prescription*), dan memberikan perkiraan (*prediction*) dari realitas yang diselidiki (Simatupang, 1984). Dalam kaitan ini, Siregar (1991) mengungkapkan bahwa suatu model yang baik memiliki karakteristik sebagai berikut:

1 Tingkat generalisasi yang tinggi

Semakin tinggi derajat generalisasi suatu model, maka ia semakin baik, sebab kemampuan model untuk memecahkan masalah semakin besar.

2 Mekanisme transparansi

Suatu model dikatakan baik jika kita dapat melihat mekanisme suatu model dalam memecahkan masalah, artinya kita bisa menerangkan kembali (*rekonstruksi*) tanpa ada yang disembunyikan. Jadi kalau ada suatu formula, maka formula tersebut dapat diterangkan kembali.

3 Potensial untuk dikembangkan

Suatu model yang berhasil biasanya mampu membangkitkan minat (*interest*) peneliti lain untuk menyelidikinya lebih jauh.

4 Peka terhadap perubahan asumsi

Hal ini menunjukkan bahwa proses pemodelan tidak pernah berakhir (selesai), selalu memberi celah untuk membangkitkan asumsi.

Melakukan eksperimen langsung pada sistem nyata untuk memahami bagaimana perilakunya, dalam beberapa keadaan, adalah sesuatu hal yang mungkin untuk dilakukan. Pada kenyataannya, keadaan sistem nyata itu terlalu kompleks, atau masih dalam bentuk hipotesis, sehingga terlalu mahal, tidak praktis atau bahkan tidak mungkin dapat dilakukan, jika harus bereksperimen langsung. Bahkan, kadang-kadang, hal itu tidak perlu. Secara umum, kendala-kendala inilah yang menjadi alasan bagi perancang untuk membuat model. Hal

ini mengkonfirmasikan lagi salah satu karakteristik model, yaitu penyederhanaan sistem nyata.

Karakteristik model sebagai penyederhanaan sistem nyata, pada ilmu pengambilan keputusan dinyatakan oleh (Guiltoni dan Martel, 1998) yang menyatakan bahwa eksperimen terbaru di bidang psikologi dan perilaku menunjukkan bahwa proses berfikir manusia tidak dapat dimodelkan dengan aturan-aturan logika dan perhitungan.

Alasan lain yang mendorong orang untuk membuat model adalah kenyataan bahwa hanya sebagian saja komponen-komponen pada suatu sistem nyata yang benar-benar menentukan perilaku sistem untuk suatu persoalan yang sedang diamati. Hal ini mengisyaratkan bahwa penggunaan model merupakan penyederhanaan masalah dengan tetap mempertahankan validitasnya.

4.1 Pengembangan Model

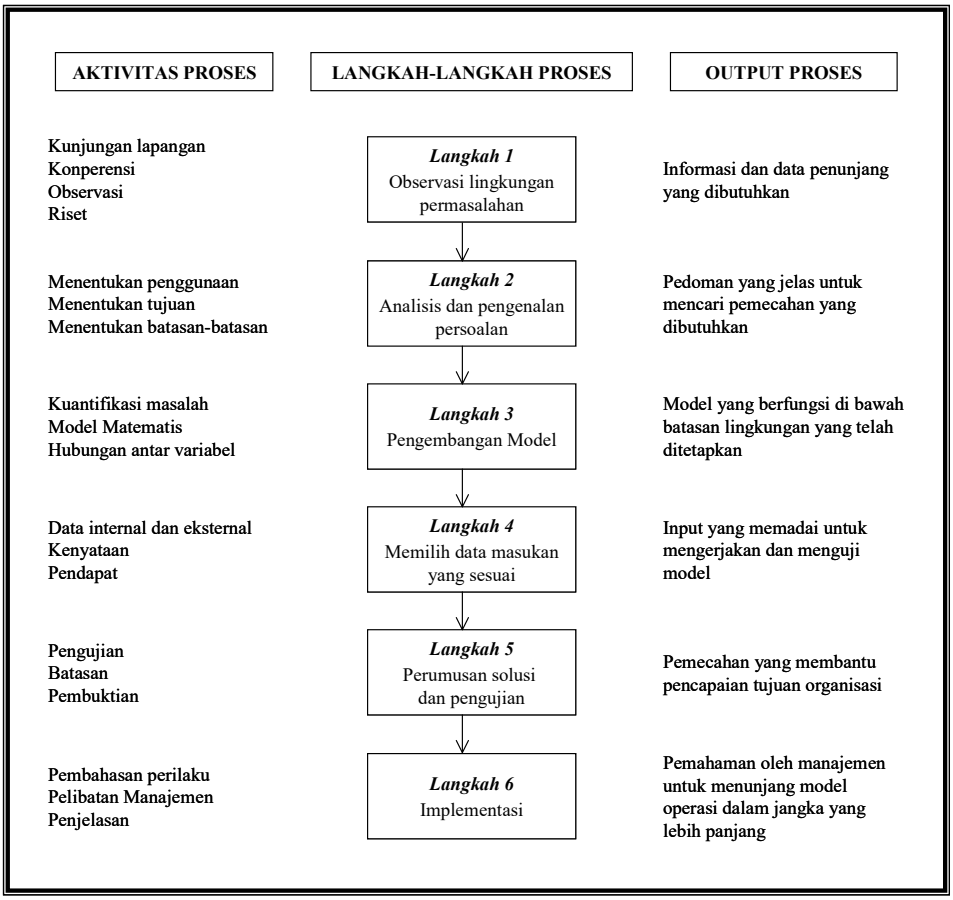
Secara umum, pengembangan model suatu sistem mengandung dua tahapan proses, yang pada prakteknya, tidak selalu harus mengikuti urutan yang diusulkan. Jadi, bisa terjadi urutan yang sebaliknya dilakukan. Kedua tahapan proses tersebut adalah sebagai berikut:

1 Pembuatan struktur model, yaitu menetapkan batas-batas sistem (*system reference*) yang akan memisahkan sistem dari lingkungannya, dan menetapkan komponen-komponen pembentuk sistem yang akan diikutsertakan atau dikeluarkan dari model. Dalam menetapkan keduanya, harus diingat bahwa model harus lengkap, valid, tetapi juga cukup sederhana.

2 Pengumpulan data, yaitu untuk mendapatkan besaran-besaran atribut komponen yang dipilih, dan untuk mengetahui hubungan yang terjadi pada aktifitas-aktifitas sistem.

4.2 Desain Model

Pengambilan keputusan dalam memecahkan masalahnya harus dibuat berdasarkan pendekatan formal tertentu. Sehingga penggunaan desain model pengambilan keputusan kerap kali menggunakan metoda kuantitatif sebagai konsep utama dalam pemecahan suatu masalah.

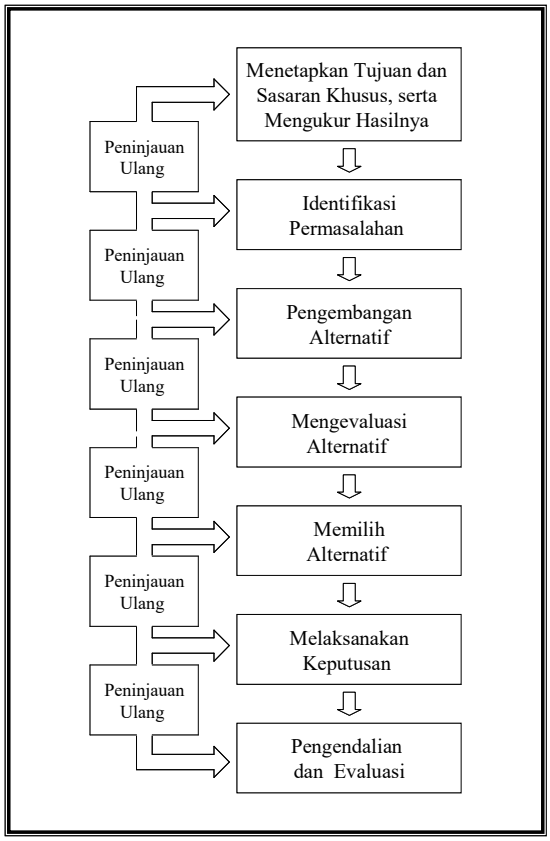


Gambar II.9 Desain Model Pengambilan Keputusan

Dalam kaitannya dengan desain model pengambilan keputusan, Gibson, et. al. (1990) mengemukakan bahwa keputusan seharusnya dipandang sebagai suatu cara dan bukannya tujuan. Keputusan merupakan *mekanisme keorganisasian*. Dengan keputusan diusahakan mencapai keadaan yang diinginkan. Keputusan itu sebenarnya merupakan tanggapan keorganisasian terhadap suatu persoalan.

Setiap keputusan adalah hasil dari proses yang dinamis yang dipengaruhi oleh kekuatan yang banyak sekali.

Proses desain ini, disajikan dalam bentuk diagram dalam Gambar II.10. Gambar tersebut menyajikan proses yang berurutan dan bukannya sebagai serangkaian langkah-langkah. Gambar ini memungkinkan diperiksanya setiap unsur dalam gerak maju yang menuju ke arah pengambilan suatu keputusan.



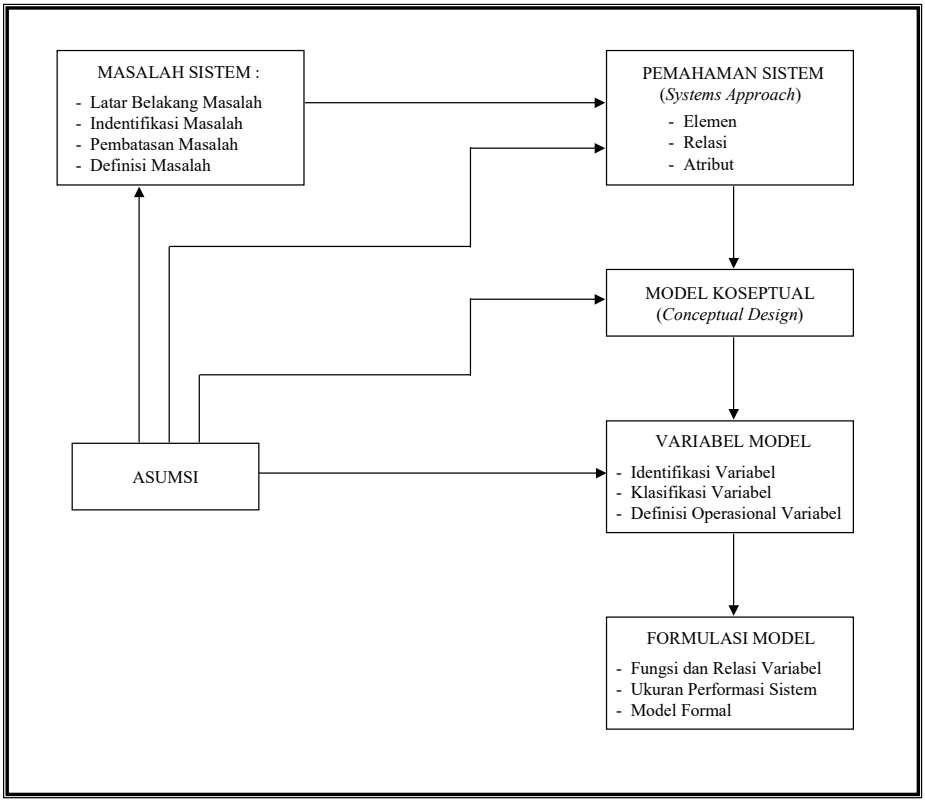
Gambar II.10 Proses Desain Model Keputusan

Berdasarkan Gambar II.10 dapat diketahui bahwa diagram ini lebih sesuai bagi keputusan yang semi terstuktur (diprogram) dibandingkan keputusan yang terstruktur. Pada persoalan yang yang timbul secara kondisional dengan tingkat ketidakpastian yang tinggi, pengambil keputusan perlu menggunakan seluruh proses. Bagi persoalan yang rutin, pengambil keputusan perlu

mempertimbangkan seluruh proses, untuk persoalan rutin ini, pengambil keputusan tidak perlu mengembangkan dan mengevaluasi setiap kali persoalan tersebut muncul.

4.3 Formulasi Model

Konsep formulasi model merupakan awal membangun model formal yang menunjukkan ukuran performansi sistem sebagai fungsi dari variabel-variabel model. Secara garis besar, langkah-langkah konsep formulasi model dapat dilihat pada Gambar II.11.



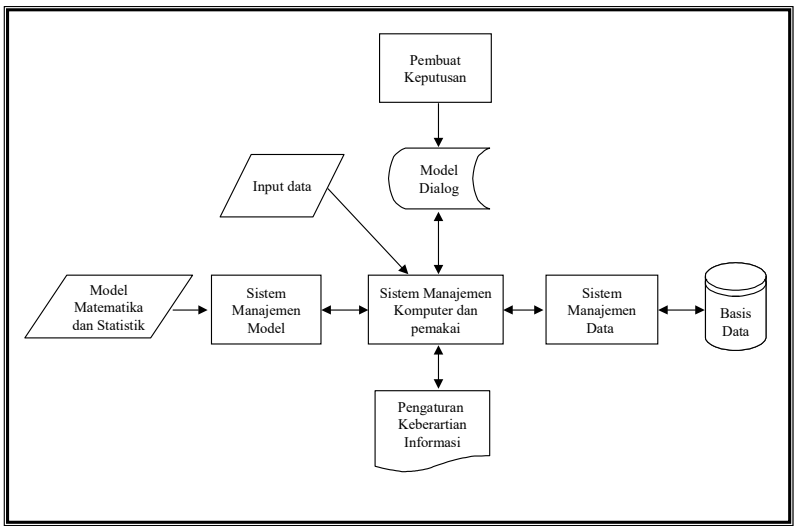
Gambar II.11 Tahap-tahap Konsep Formulasi Model (Simatupang, 1994)

Model merupakan cara sederhana untuk memandang suatu masalah. Model yang baik cukup hanya mengandung bagian-bagian yang yang perlu saja. Untuk memudahkan pemikiran tentang karakteristik-karakteristik model yang dibuat, haruslah kita mengerti tentang masalah (problem) dan sistemnya.

Dalam pembentukan model, harus diperhatikan faktor apa saja yang harus mempengaruhi prilaku dari sistemnya, atau dengan kata lain harus memperhatikan pengertian (konsep) sistemnya. Dengan demikian, dapat ditentukan variabel-variabel apa saja yang dapat menentukan performansi dari sistem yang diamati, kemudian bagaimana variabel-variabel tersebut dapat dikendalikan dan diatur. Pada akhirnya akan diperoleh suatu performansi sistem yang dikehendaki.

Ada beberapa kriteria yang harus dipenuhi dalam memodelkan suatu sistem, antara lain: (a) model harus mewakili (merepsentasikan) sistem nyatanya; dan (b) model merupakan penyederhanaan dari kompleksnya sistem, sehingga diperbolehkan adanya penyimpangan pada batas-batas tertentu.

Model tidak hanya digunakan untuk menggambarkan sekumpulan pemikiran-pemikiran, tetapi juga mengadakan evaluasi dan meramalkan kelakuan sistem, sehingga akan didapatkan perancangan terbaik tanpa membutuhkan konstruksi seluruh kenyataan alamiahnya. Bentuk integrasi model dalam SPK dapat dijelaskan melalui Gambar II.12.



Gambar II.12 Model Matematis/Statistik dalam SPK (Thierauf, 1982).

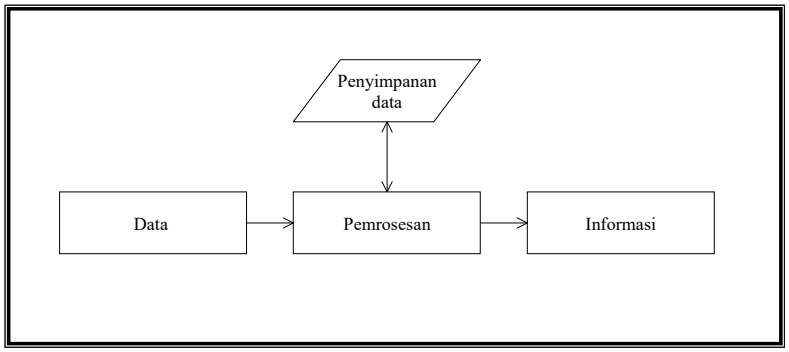
Kebanyakan masalah yang dihadapi oleh pengambil keputusan adalah belum dimilikinya definisi atau susunan sistem yang jelas. Jadi harus dilakukan pendekatan sistem untuk membangun sistemnya secara eksplisit. Lagi pula, sering masalah yang dihadapi merupakan masalah yang unik yang bisa saja terjadi dengan latar belakang yang berbeda. Memang telah banyak model yang tersedia yang tampaknya cocok dengan masalah yang sedang dihadapi, namun pertanyaan klasik selalu muncul yakni, bagaimana caranya memakai model tersebut. Dengan kata lain apa yang harus dilakukan agar model yang ada dapat dipakai tanpa mengurangi nilai pemecahan masalah. Oleh karena itu, diperlukan modifikasi dan pengembangan model dari sistem masalah yang ditinjau. Pengembangan model tidak lain adalah suatu usaha memperoleh model baru yang memiliki kemampuan lebih di dalam beberapa aspek.

4.4 Aspek Informasi

Dalam suatu sudut pandang, Mangkusubroto dan Tresnadi (1987) menyatakan bahwa informasi merupakan suatu produk komunikasi yang memberi pengaruh pada meningkatnya pengetahuan seseorang terhadap suatu hal. Dalam persoalan keputusan, informasi ini berkaitan erat dengan ketidakpastian yang melingkup variabel-variabel persoalan tersebut; dimana untuk mengurangi derajat ketidakpastian ini diperlukan informasi tambahan.

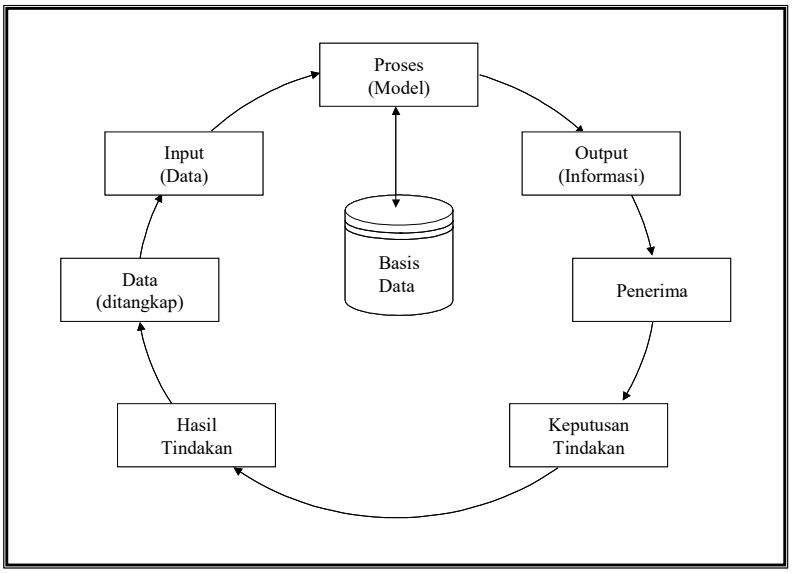
4.4.1 Data dan Informasi

Informasi dan data adalah dua hal yang berbeda. Walaupun demikian, keduanya berkaitan erat dengan fakta. Davis (1974) mendefinisikan data sebagai bahan informasi yang dirumuskan sebagai kumpulan dari simbol-simbol yang teratur, yang dinyatakan sebagai jumlah, tindakan-tindakan, hal-hal dan sebagainya. Data-data dibentuk dari lambang grafis, alphabetis, numerik, atau lambang khusus. Sedangkan informasi adalah data yang telah diolah ke dalam bentuk yang berarti bagi pengambil keputusan, mempunyai nilai guna/manfaat dalam proses pengambilan keputusan.



Gambar II.13 Proses Transformasi Data

Jogiyanto (1995) menyatakan bahwa aliran data menjadi informasi, yang kemudian dijadikan sebagai referensi pengambilan keputusan, pada prinsipnya membentuk suatu siklus informasi. Siklus ini disebut juga sebagai siklus pengolahan data.



Gambar II.14 Siklus Informasi (Jogiyanto, 1995).

4.4.2 Sumber Informasi

Mangkusubroto dan Tresnadi (1987) mengungkapkan bahwa pada dasarnya informasi dapat dibedakan menjadi dua, yaitu data empiris dan informasi dari ahli.

a Data Empiris

Data empiris yang diperoleh melalui suatu pengumpulan data atau survei dapat digunakan untuk menaksir distribusi kemungkinan munculnya suatu kejadian. Dalam hal ini, pendapat awal (atau sering juga dinyatakan sebagai nilai kemungkinan prior) digunakan untuk menguji bahwa frekuensi relatif tersebut mencerminkan nilai kemungkinan pengambil keputusan. Data empiris ini dapat pula digunakan untuk mendapatkan distribusi kemungkinan akhir (*posterior*) berdasarkan nilai kemungkinan prior yang telah ada.

b Informasi dari ahli

Dalam beberapa hal, karena terbatasnya pengetahuan, waktu, dan lain-lain, data empiris sulit sekali diperoleh. Dalam keadaan seperti ini maka satu-satunya sumber informasi adalah pendapat atau pandangan subjektif dari ahli atau orang yang lebih mengetahui tentang kejadian yang tidak pasti tersebut.

4.4.3 Nilai Informasi

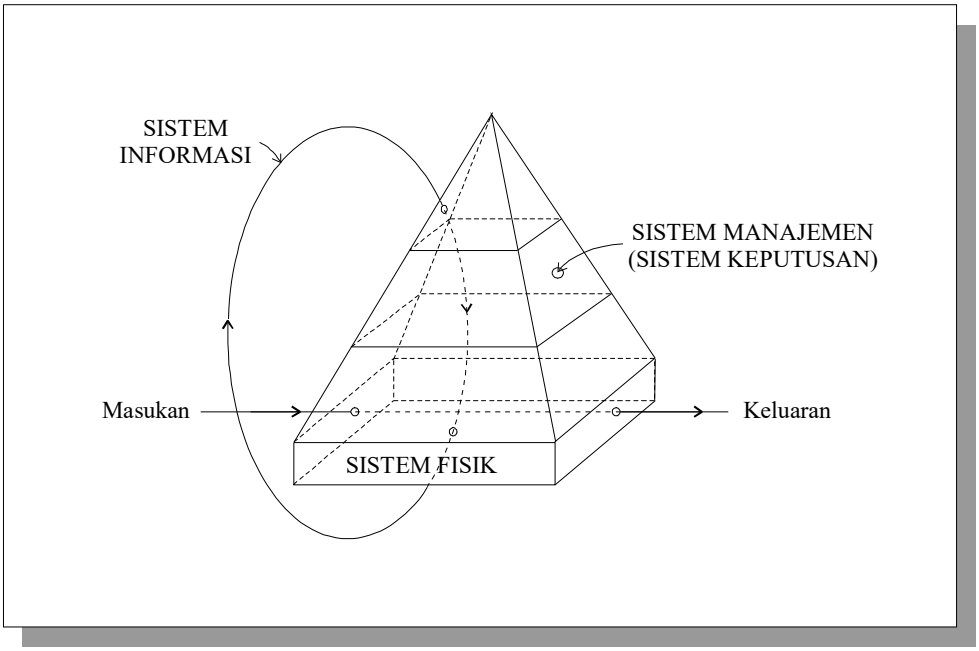
Pertanyaan mendasar berkenaan dengan nilai informasi adalah; "Berapakah nilai suatu informasi untuk suatu persoalan keputusan tertentu ?". Prinsip utama yang berkenaan dengan hal ini adalah bahwa informasi hanya mempunyai nilai bila informasi tersebut dapat mengakibatkan suatu perubahan dalam tindakan yang diambil oleh pengambil keputusan. Meskipun suatu data atau pernyataan seorang ahli dapat memberikan pengetahuan baru, namun hal ini tidak akan mempunyai nilai dalam konteks suatu keputusan tertentu, selama informasi tambahan tersebut tidak dapat mengakibatkan perubahan dalam sikap maupun tindakan berkenaan dengan persoalan yang dihadapi (Mangkusubroto dan Tresnadi, 1987).

Selanjutnya, Jogiyanto (1995) menyatakan bahwa nilai dari informasi ditentukan oleh dua hal, yaitu manfaat dan biaya. Suatu informasi dikatakan bernilai bila manfaatnya lebih besar dibandingkan dengan biaya untuk

memperolehnya. Pengukuran nilai informasi biasanya dihubungkan dengan dengan analisis *cost effectiveness* atau *cost-benefit*.

5 Pengambilan Keputusan dalam Organisasi

Sistem pengambilan keputusan merupakan bagian tak terpisahkan dari totalitas sistem organisasi keseluruhan. Dalam gambar ini diperlihatkan bahwa sistem organisasi paling tidak mencakup sistem fisik (sistem operasional), sistem manajemen (sistem keputusan) dan sistem informasi (Suryadi, 1992).



Gambar II.15 Posisi Sistem Keputusan dalam Sistem Organisasi (Suryadi, 1992)

Sistem fisik (sistem operasional) mencerminkan proses transformasi dari input (masukan) menjadi output (keluaran) melalui serangkaian mekanisme/proses dengan melibatkan sumber daya manusia dan non manusia (mesin, uang, bahan baku, energi, informasi, dan lain-lain). Semakin besar aktifitas sistem fisik, maka semakin kompleks permasalahan yang dihadapi. Hal ini mengakibatkan semakin rumitnya proses pengaturan yang dihadapi.

Bertitik tolak dari pemikiran diatas, maka kelancaran sistem fisik sangat dipengaruhi oleh mekanisme pengaturan yang dijalani. Rangkaian pengaturan sistem fisik ini distrukturkan dalam sistem manajemen yang tidak lain merupakan sistem yang menghasilkan keputusan-keputusan yang diperlukan guna menjamin kelancaran sistem fisik. Oleh karena sistem manajemen ini menghasilkan sejumlah keputusan, maka sering pula sistem manajemen ini disebut sebagai sistem keputusan.

Berdasarkan uraian di atas, maka sistem keputusan tidak bisa dipisahkan dari sistem fisik maupun sistem informasi. Kompleksitas sistem fisik menuntut adanya sistem keputusan yang kompleks pula. Untuk pemecahan masalah yang kompleks ini diperlukan suatu model pengambilan keputusan yang menggunakan instrumen metodologik yang mampu mengakomodasi masalah yang multikompleks dengan begitu banyak pihak terkait yang masing-masing mempunyai persepsi dan kepentingan yang berbeda.

Perencanaan Strategis

Perencanaan strategis adalah aktivitas tertinggi yang akan menentukan arah gerak dan perkembangan organisasi dalam suatu periode waktu yang panjang. Melalui aktivitas ini, sasaran-sasaran, kebijakan-kebijakan, dan pedoman umum. Dalam melaksanakan perencanaan strategis ini perkembangan terhadap keadaan lingkungan sangat memegang peranan, disamping kondisi organisasi yang merupakan dasar dan titik tolaknya. Oleh sebab itu, informasi yang dibutuhkan haruslah memberikan gambaran yang menyeluruh dan lengkap, walaupun tidak harus memiliki ketelitian yang tinggi.

Dalam teori manajemen strategis, pengorganisasian yang fleksibel diperhitungkan sebagai aset strategis dalam situasi yang serba mengejutkan (Volberda dan Rutges, 1999).

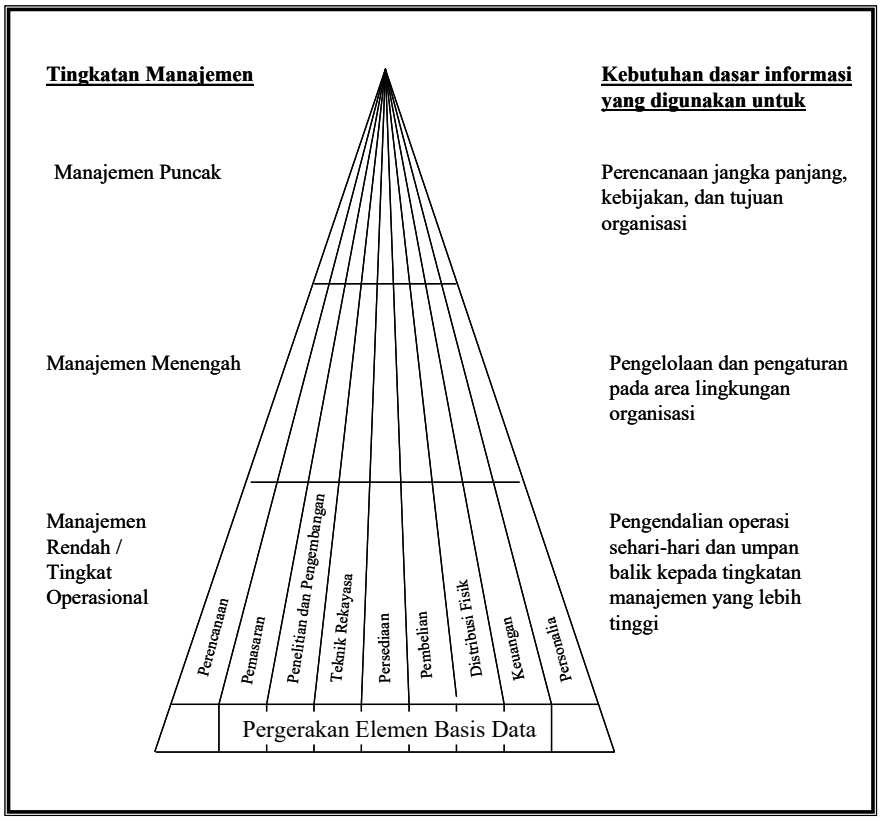
Kinerja yang baik pada suatu organisasi adalah hasil dari perbaikan interaksi yang benar pada manajemen organisasi dengan lingkungan internal dan eksternalnya. Analisis atas lingkungan internal (kekuatan dan kelemahan) dan lingkungan eksternal (peluang dan ancaman) merupakan basis dalam perencanaan strategis (Houben et. al., 1999).

Pengendalian Manajemen

Pengendalian manajemen adalah aktivitas pengendalian yang sifatnya lebih luas dari pengendalian operasi. Ia bertugas mengawasi pelaksanaan program yang ada dalam usaha untuk memperbaiki pelaksanaan program tersebut, untuk pencapaian tujuan.

Pengendalian Operasional

Pengendalian operasional adalah aktivitas pendayagunaan fasilitas dan sumberdaya yang ada untuk penyelenggaraan kegiatan-kegiatan operasi. Informasi yang diperlukan memerlukan tingkat ketelitian, tingkat ketepatan dan tingkat keterperincian yang tinggi.



Gambar II.16 Struktur informasi Vertikal dan Horizontal pada Basis Data Komprehensif (Thierauf, 1982).

Banyak riset tambahan diperlukan untuk mengembangkan pengetahuan tentang komponen efektivitas. Dalam hal ini, Gibson et. al. (1990) menyatakan bahwa di antara para ahli hanya ada sedikit kesepakatan mengenai komponen yang relevan dan juga mengenal hubungan timbal balik antara komponen-komponen itu dan mengenai pengaruh tindakan manajemen terhadap komponen tersebut.

6 Sistem Pendukung Keputusan

SPK adalah perangkat pendukung keputusan yang berbasiskan komputer untuk membantu pengambil keputusan dengan menyajikan informasi dan interpretasinya untuk berbagai alternatif keputusan (Pal dan Palmer, 2000)

Ruang lingkup SPK menurut Radermacher (1994) terdiri dari tiga bagian, yaitu sistem, pendukung, dan keputusan. Aspek sistem merupakan otomatisasi yang

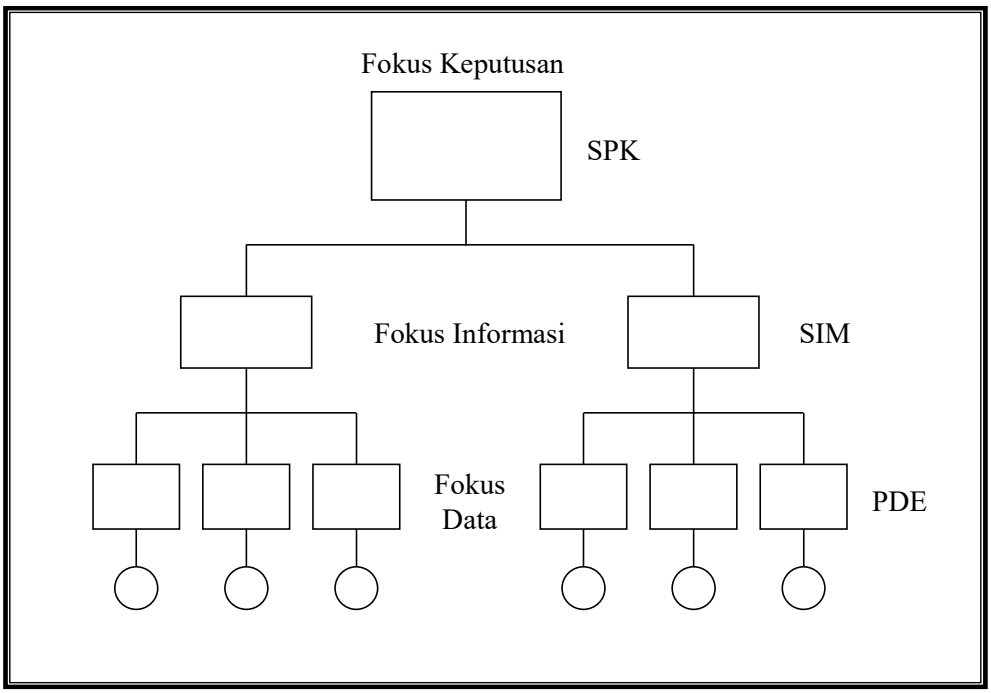
membantu pengambil keputusan dengan tingkat kecerdasan yang berbeda. Berdasarkan sudut pandang teoritis, keputusan berhubungan dengan konsep kognitif, terutama yang berkaitan dengan ide yang mendukung manusia dalam pembuatan keputusan.

SPK dapat membantu pengambil keputusan dalam membuat keputusan strategis. Penggunaan SPK telah menunjukkan hasil yang memuaskan, dengan meminimalkan biaya, percepatan proses pengambilan keputusan, dan hasil yang signifikan dalam keunggulan bersaing. Pada sisi lain, Karacapilidis dan Pappis (1997) menyatakan bahwa kelebihan SPK adalah kemampuannya dalam mengintegrasikan sistem interaktif untuk para manager dan kaum professional, lingkungan yang bersahabat dengan pengguna (*user-friendly*), dan penyediaan kerangka yang memadai untuk mengatasi permasalahan yang semi terstruktur.

Dua pandangan yang menunjukkan perbedaan antara Sistem Pendukung Keputusan dengan Sistem Informasi Manajemen, yaitu sudut pandang konotasional dan teoritikal (Sprague dan Carlson, 1982).

Sudut Pandang Konotasinal

Dalam sudut pandang konotasinal, SPK adalah kemajuan secara revolusioner dari SIM dan PDE. Penggambaran jenis ini, dapat dilihat pada Gambar II.17.



Gambar II.17 SPK dalam Sudut Pandang Konotasional (Sprague dan Carlson, 1982)

PDE diterapkan pada level operasional organisasi. Karakteristik PDE meliputi aktivitas-aktivitas:

- Menitikberatkan pada data, penyimpanan, pengolahan, dan aliran pada level operasional
- Membantu pengolahan transaksi-transaksi secara lebih efisien
- Memungkinkan pengolahan komputer secara lebih terjadwal dan optimum
- Menyediakan pembukuan (file) terpadu untuk kegiatan yang saling berkaitan
- Memberikan laporan umum atau ikhtisar kepada manajemen

Dengan adanya peningkatan kemampuan dan kecepatan piranti keras, sistem operasi on line, pilihan komunikasi data yang menarik dan kemampuan penuh dari terminal, aktivitas pada level EDP ini menjadi lebih lancar dan lebih efisien dalam penggunaan fasilitas dalam pengolahan data transaksi.

Secara umum SIM difokuskan pada tingkat yang lebih tinggi dalam organisasi. SIM memiliki karakter sebagai berikut:

- Menitikberatkan pada informasi bagi para manager menengah
- Menangani aliran-aliran informasi yang terstruktur
- Memadukan PDE dari kegiatan-kegiatan berdasarkan fungsi usaha (SIM Pemasaran, SIM Produksi, dan lain-lain).
- Melayani kebutuhan informasi dan pembuatan laporan, umumnya melalui suatu basis data

Dari karakteristik diatas, terlihat bahwa Sistem Informasi Manajemen berorientasi pada struktur aliran informasi dan operasional (rutinitas).

SPK, menurut tinjauan konotatif ini, merupakan sistem yang ditujukan kepada tingkatan manajemen yang lebih tinggi lagi, dengan penekanan karakteristik sebagai berikut:

- Berfokus pada keputusan, ditujukan pada manajer puncak dan pengambil keputusan
- Menekankan pada fleksibilitas, adaptabilitas dan respon yang cepat
- Mampu mendukung berbagai gaya pengambilan keputusan dari masing-masing pribadi manajer.

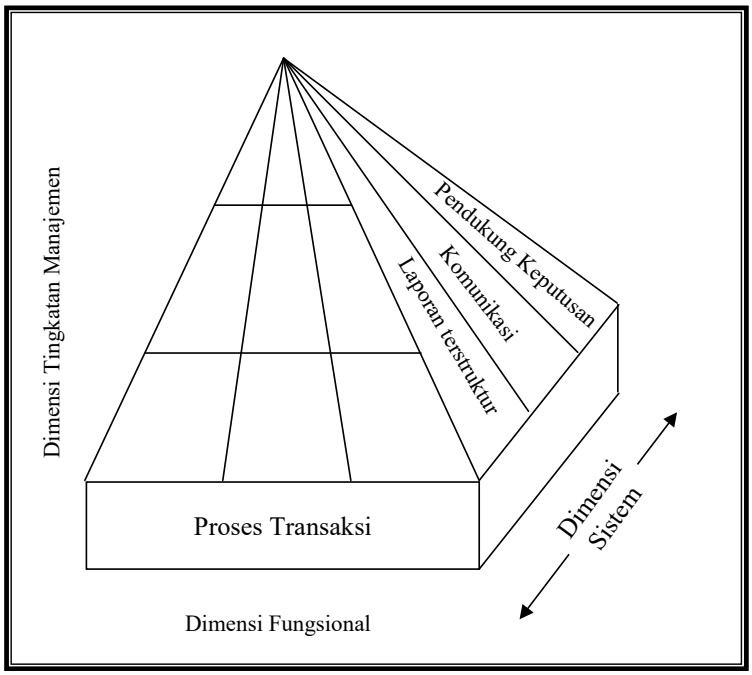
Namun, dalam hal ini terdapat beberapa kelemahan pada pandangan konotasi, antara lain:

- Adanya gambaran bahwa SPK seakan-akan hanya dibutuhkan pada tingkat manajemen puncak. Pada kenyataannya, dukungan bagi pengambilan keputusan dibutuhkan pada semua tingkatan manajemen dalam suatu organisasi.
- Pengambilan keputusan yang terjadi pada beberapa level harus dikoordinasikan. Jadi dimensi dari pendukung keputusan adalah

komunikasi dan koordinasi diantara pengambil keputusan antar level organisasi yang berbeda maupun pada level organisasi yang sama.

Sudut Pandang Teoritikal

Tujuan fungsional sistem informasi dalam suatu organisasi adalah untuk meningkatkan prestasi kerja karyawan dalam organisasi melalui penerapan teknologi informasi.



Gambar II.18 SPK dalam Sudut Pandang Teoritikal (Sprague dan Carlson, 1982).

Piramid ini dikembangkan oleh Robert Head pada akhir tahun 1960-an sebagai model visual untuk mencirikan SIM dalam pengertian yang luas. Dimensi vertikal menggambarkan level manajemen dan dimensi horizontal menggambarkan unit-unit fungsional utama dalam kegiatan organisasi. Dimensi bagian dalam menunjukkan subsistem teknologi yang memberi dukungan bagi aktivitas manajemen.

Jadi, SPK bukan sekedar pengembangan evolusioner dari PDE dan SIM, namun SPK merupakan kelas sistem informasi yang berinteraksi dengan bagian-bagian lain dari sistem informasi manajemen secara keseluruhan untuk mendukung aktivitas pengambilan keputusan dalam organisasi. Kajian SPK merupakan studi pendalaman atas keragaman sistem, perangkat, dan teknologi, dengan fokus inti dari SPK adalah optimasi dan model simulasi (Power, 1998).

Muncul dugaan bahwa SPK dapat meningkatkan kualitas keputusan, sejak itu dilakukan beberapa penelitian yang memberikan fokus untuk mengkaji efektivitas dan efisiensi penggunaan SPK. Hasil penelitian-penelitian ini memberikan hasil berupa indikasi bahwa penggunaan SPK memberikan hasil yang tidak pasti. Pada beberapa kasus kualitas keputusan meningkat, pada sisi lain dijumpai kualitas keputusan yang mengalami penurunan, dan juga dijumpai kenyataan bahwa penggunaan SPK tidak memberikan pengaruh apa-apa. Penelitian-penelitian ini mengemukakan temuan bahwa ada beberapa variabel, masalah, dan hubungan interaksi ketika menilai penggunaan SPK.

Faktor-faktor tersebut dikategorikan sebagai faktor SPK, faktor tugas, dan karakteristik pengguna. Sebagai contoh, faktor SPK adalah munculnya sugesti atas efektivitas dan efisiensi SPK memiliki pengaruh yang kontradiktif dengan penggunaan SPK. Faktor tugas adalah jumlah tugas seseorang yang akan meningkat dengan penggunaan SPK sebagai akibat meningkatnya jumlah alternatif pilihan. Contoh dari karakteristik pengguna terdiri dari pengalaman, preferensi data, dan usaha. Interaksi dari ketiga faktor tersebut berpengaruh terhadap efektivitas penggunaan SPK (Kanungo et. al., 2001).

Pada dasarnya SPK dirancang untuk mendukung seluruh tahap pengambilan keputusan mulai dari identifikasi masalah, pemilihan data yang relevan, menentukan pendekatan yang digunakan dalam proses pengambilan keputusan, sampai evaluasi pemilihan alternatif. Thierauf (1982), menyatakan terdapat sepuluh karakteristik dasar SPK yang efektif, yaitu:

a Mendukung proses pengambilan keputusan yang menitikberatkan pada *management by perception*

b Interface manusia/mesin dimana manusia (*user*) tetap mengontrol proses pengambilan keputusan

c Mendukung pengambilan keputusan untuk membahas masalah-masalah terstruktur dan semi terstruktur.

d Menggunakan model-model matematis dan statistik yang sesuai.

e Memiliki kapabilitas pertanyaan untuk memperoleh informasi sesuai dengan kebutuhan - *mode interaktif*

f Output ditujukan untuk personil organisasi dalam semua tingkatan.

g Memiliki subsistem-subsistem yang terintegrasi dalam suatu SPK secara sedemikian rupa sehingga dapat berfungsi sebagai kesatuan sistem

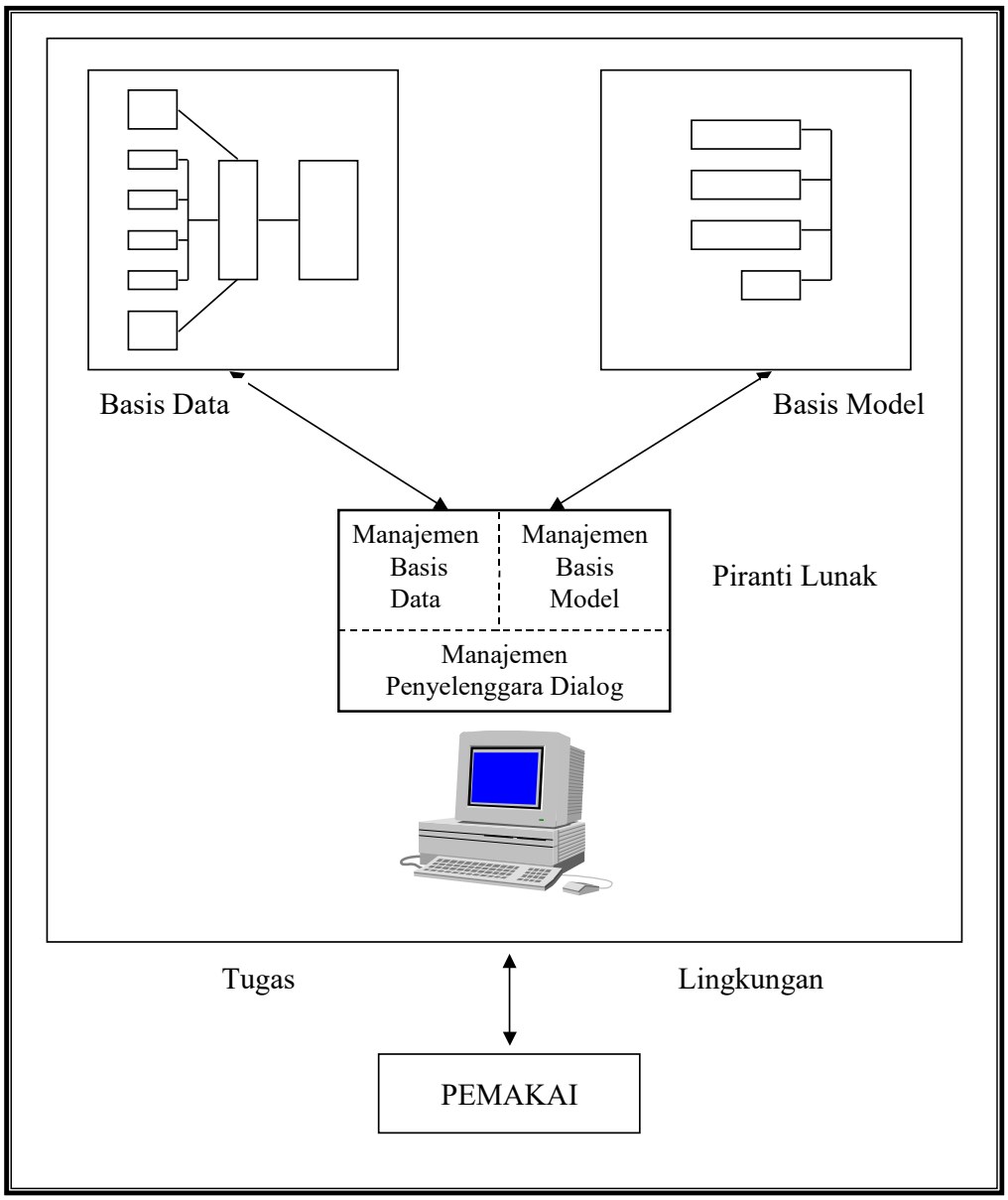
h Membutuhkan struktur data komprehensif yang dapat melayani kebutuhan informasi seluruh tingkatan manajemen

i Pendekatan *easy to use*. Ciri suatu SPK yang efektif adalah kemudahannya untuk digunakan, dan memungkinkan kebebasan pemakai untuk memilih atau mengembangkan pendekatan-pendekatan baru untuk membahas masalah yang dihadapi.

j Kemampuan sistem beradaptasi secara cepat, dimana pengambil keputusan dapat menghadapi masalah-masalah baru, dan pada saat yang sama menanganinya dengan cara mengadaptasikan sistem terhadap kondisi-kondisi perubahan yang terjadi.

6.1 Komponen-Komponen SPK

Secara umum, SPK memiliki tiga subsistem utama yang menentukan kapabilitas teknis SPK tersebut, yaitu subsistem manajemen basis data, subsistem manajemen basis model, dan subsistem perangkat lunak penyelenggara dialog (Sprague dan Carlson, 1982).

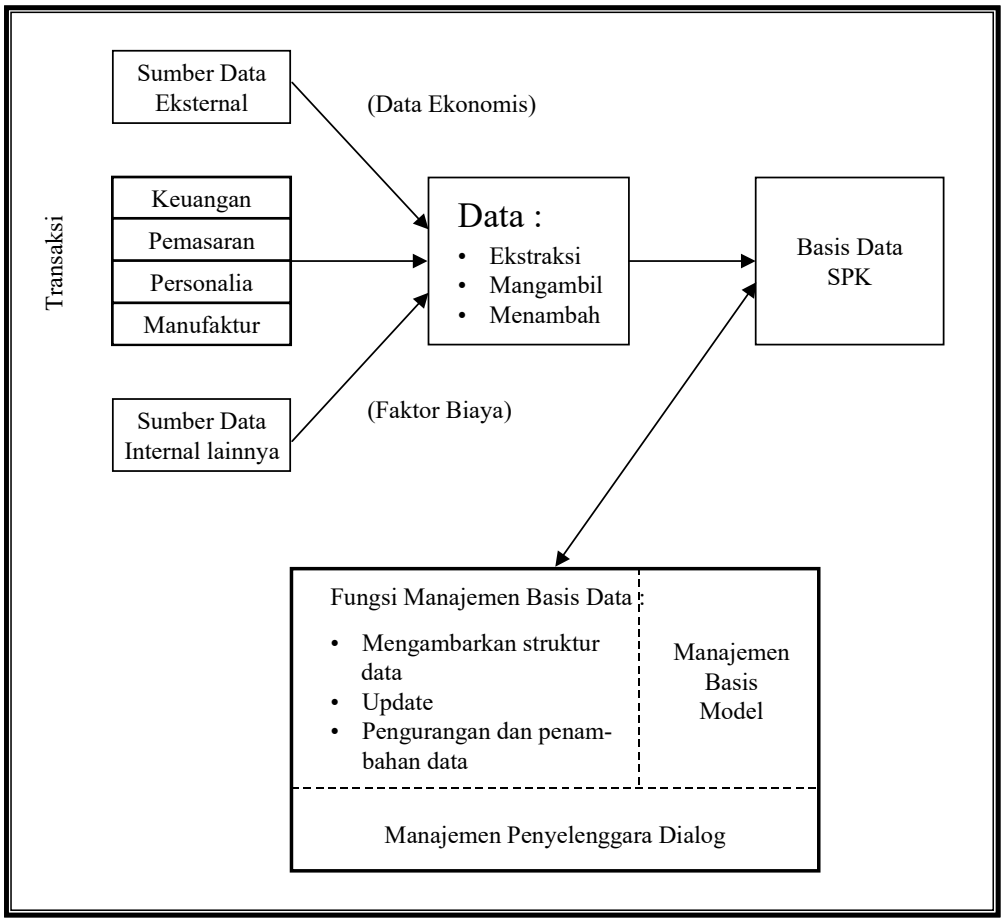


Gambar II.19 Komponen Sistem Pendukung Keputusan (Sprague dan Carlson, 1982)

Sedangkan Power (2000) menyatakan bahwa SPK terdiri atas empat komponen utama, yaitu (1) interface pengguna, (2) basis data, (3) model dan perangkat analisa, dan (4) jaringan komunikasi.

Subsistem Manajemen Basis Data (*Data Base Management Subsystem/ DBMS*)

Manajemen basis data adalah pengendali basis data dari koleksi dari struktur data yang umum dan historis, yang diperoleh dari berbagai sumber yang telah diorganisasikan untuk memudahkan proses akses dan analisis (Power, 2000).



Gambar II.20 Subsistem Manajemen Basis Data (Sprague dan Carlson, 1982)

Subsistem manajemen basis data menyediakan fasilitas untuk menyimpan data, permintaan (*request*), pemanggilan (*retrieval*), dan pemeliharaan untuk seluruh data sistem (Matsatsinis dan Siskos, 1999). Subsistem data yang tercakup dalam DBMS. Ada beberapa perbedaan antara basis data untuk SPK dan non-SPK. Pertama, sumber data untuk SPK lebih "kaya" dari pada non SPK dimana data harus berasal dari luar dan dari dalam, karena proses pengambilan

keputusan, terutama dalam level manajemen puncak, sangat tergantung pada sumber data dari luar seperti data ekonomi.

SPK membutuhkan proses ekstraksi, dan DBMS yang mengelolanya harus cukup fleksibel untuk memungkinkan penambahan dan pengurangan secara cepat. Dalam hal ini, kemampuan yang dibutuhkan dari manajemen basis data dapat diringkas, sebagai berikut:

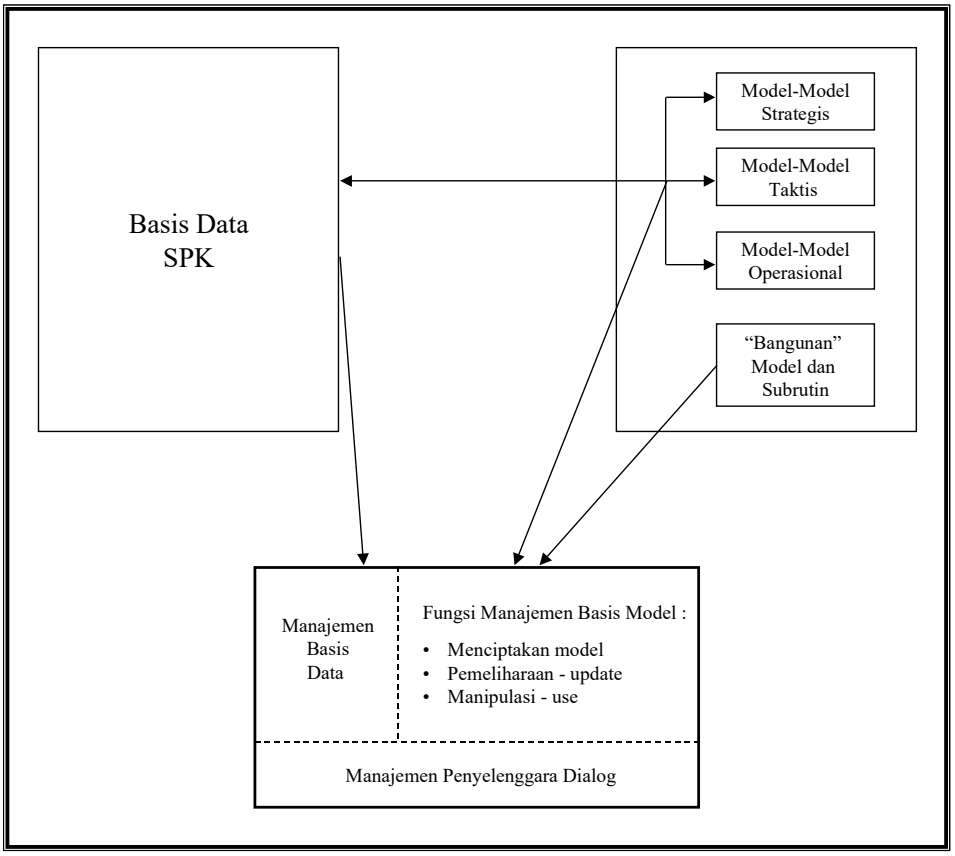
- Kemampuan untuk mengkombinasikan berbagai variasi data melalui pengambilan dan ekstraksi data
- Kemampuan untuk menambahkan sumber data secara cepat dan mudah
- Kemampuan untuk menggambarkan struktur data logikal sesuai dengan pengertian pemakai sehingga pemakai mengetahui apa yang tersedia dan dapat menentukan kebutuhan penambahan dan pengurangan
- Kemampuan untuk menangani data secara personil sehingga pemakai dapat mencoba berbagai alternatif pertimbangan personil.
- Kemampuan untuk mengelola berbagai variasi data.

Subsistem Manajemen Basis Model (*Model Base Management Subsystem/ MBMS*)

Salah satu keunggulan SPK adalah kemampuan untuk mengintegrasikan akses data dan model-model keputusan. Hal ini dapat dilakukan dengan menambahkan model-model keputusan ke dalam sistem informasi yang menggunakan basis data sebagai mekanisme integrasi dan komunikasi di antara model-model. Karakteristik ini menyatukan kekuatan pencarian dan pelaporan data dari PDE dan pengembangan disiplin manajemen. SPK mendukung proses pengambilan keputusan melalui penggunaan model-model matematik (Raghunathan, 1996).

Salah satu persoalan yang berkaitan dengan model adalah bahwa penyusunan model seringkali terikat pada struktur model yang mengasumsikan adanya masukan yang benar dan cara keluaran yang tepat. Sementara itu, model

cenderung tidak mencukupi karena adanya kesulitan dalam mengembangkan model yang terintegrasi untuk menangani sekumpulan keputusan yang saling bergantungan. Cara untuk menangani persoalan ini adalah dengan menggunakan koleksi berbagai model yang terpisah, dimana setiap model digunakan untuk menangani bagian yang berbeda dari masalah yang sedang dihadapi. Komunikasi antara berbagai model digunakan untuk menangani bagian yang berbeda dari masalah tersebut. Komunikasi antara berbagai model yang saling berhubungan diserahkan kepada pengambil keputusan sebagai proses intelektual dan manual.



Gambar II.21 Subsistem Manajemen Basis Model (Sprague dan Carlson, 1982)

Salah satu pandangan yang lebih optimis berharap untuk bisa menambahkan model-model ke dalam sistem informasi dengan basis data sebagai meknisme

integrasi dan komunikasi di antara mereka. Kemampuan yang dimiliki subsistem basis model meliputi:

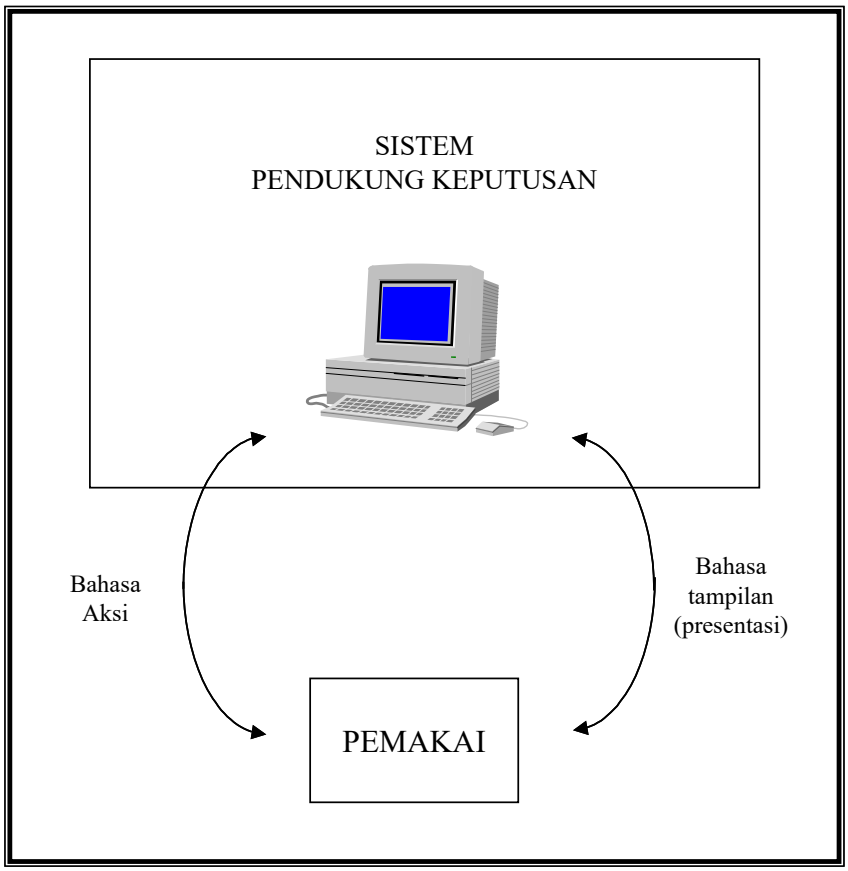
- Kemampuan untuk menciptakan model-model baru secara cepat dan mudah
- Kemampuan untuk mengakses dan mengintegrasikan model-model keputusan
- Kemampuan untuk mengelola basis model dengan fungsi manajemen yang analog dan manajemen basis data (seperti mekanisme untuk menyimpan, membuat dialog, menghubungkan, dan mengakses model).

Model matematik dan model analisis adalah komponen terbesar dari pengendali model SPK. Setiap pengendali model SPK memiliki tujuan yang spesifik. Pemilihan model yang layak adalah kunci performasi rancangan SPK. Dalam pengendali model SPK, nilai kunci dari variabel atau parameter dirubah untuk menghasilkan suatu informasi. Informasi dari model kemudian dianalisa dan dievaluasi oleh pembuat keputusan (Power, 2000).

Subsistem Perangkat Lunak Penyelenggara Dialog (*Dialog Generation and Management Software*)

Fleksibilitas dan kekuatan karakteristik SPK timbul dari kemampuan interaksi antara sistem dan pemakai, yang dinamakan subsistem dialog. Kemampuan interaksi ini merupakan salah satu kunci sukses dari penggunaan SPK (Jones, 1995). Subsistem dialog menyajikan bentuk komunikasi antara pengguna dengan sistem, kemampuan yang disajikan subsistem dialog meliputi penyajian bahasa komunikasi yang mudah, pesan yang jelas, dan penyajian dialog secara grafis (Matsatsinis daan Siskos, 1999).

Bennet mendefinisikan pemakai, terminal, dan sistem perangkat lunak sebagai komponen-komponen dari sistem dialog. Ia membagi subsistem dialog menjadi tiga bagian (Gambar II.22):



Gambar II.22 Subsistem Penyelenggaraan Dialog (Sprague dan Carlson, 1982)

a *Bahasa aksi*, meliputi apa yang dapat digunakan oleh pemakai dalam berkomunikasi dengan sistem. Hal ini meliputi pemilihan-pemilihan seperti papan ketik (*keys board*), panel-panel sentuh, joystick, perintah suara dan sebagainya.

b *Bahasa tampilan* atau *presentasi*, meliputi apa yang harus diketahui oleh pemakai. Bahasa tampilan meliputi pilihan-pilihan seperti printer, layar tampilan, grafik, warna, plotter, keluaran suara, dan sebagainya.

c *Basis Pengetahuan*, meliputi apa yang harus diketahui oleh pemakai. Basis pengetahuan meliputi apa yang harus diketahui oleh pemakai agar pemakaian sistem bisa efektif. Basis pengetahuan bisa berada dalam pikiran pemakai, pada kartu referensi atau petunjuk, dalam buku manual, dan sebagainya.

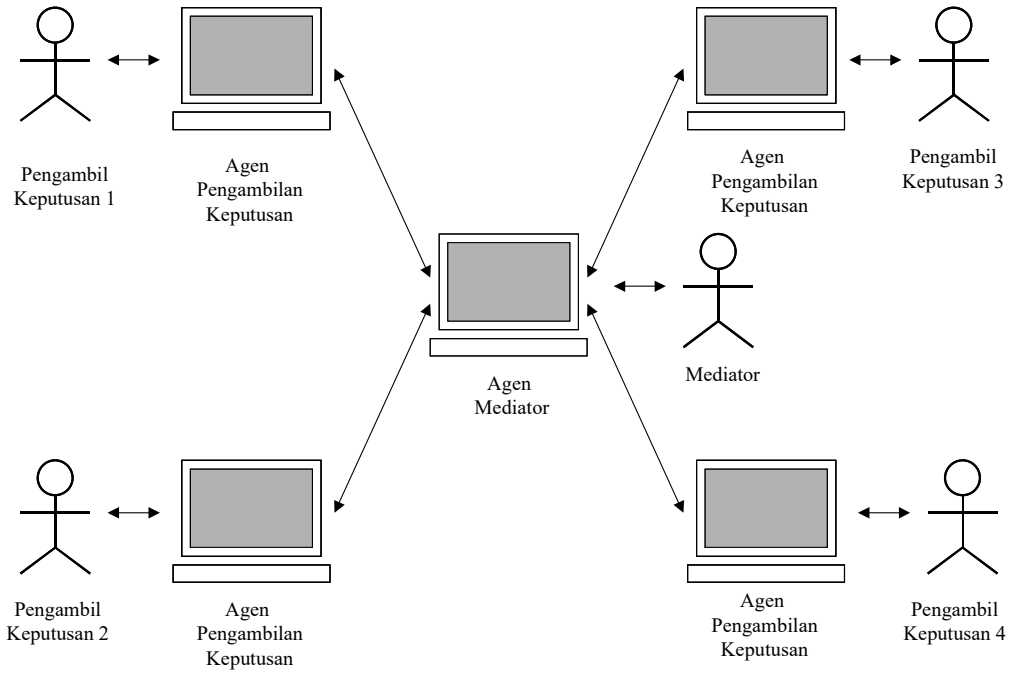
Kombinasi dari kemampuan-kemampuan diatas terdiri dari apa yang disebut gaya dialog. Misalnya yang meliputi pendekatan tanya dan jawab, bahasa perintah, menu-menu, dan mengisi tempat kosong.

Kemampuan yang harus dimiliki oleh SPK untuk mendukung dialog pemakai/sistem meliputi:

- Kemampuan untuk menangani berbagai variasi gaya dialog, bahkan jika mungkin untuk mengkombinasikan berbagai gaya dialog sesuai dengan pilihan pemakai.
- Kemampuan untuk mengakomodasi tindakan pemakai dengan berbagai peralatan masukan
- Kemampuan untuk menampilkan data dengan berbagai variasi format dan peralatan keluaran.
- Kemampuan untuk memberikan dukungan yang fleksibel untuk mengetahui basis pengetahuan pemakai.

Subsistem Jaringan Komunikasi

Perkembangan lebih lanjut dari SPK adalah SPK Kelompok (*Group Decision Support System*). SPK Kelompok adalah sistem interaktif yang berbasis komputer yang memiliki fasilitas untuk mengatasi masalah yang semi terstruktur, yang diselesaikan oleh sejumlah pengambil keputusan yang bekerja dalam sebuah tim. Tujuan utama dari SPK Kelompok adalah penciptaan keputusan kelompok yang efektif dengan proses pembagian informasi yang interaktif diantara anggota kelompok. Hal ini menyangkut tentang penyediaan teknik untuk membuat struktur analisa keputusan, teknik analisa keputusan, penjadwalan, dan proses komunikasi jarak jauh (Karacapilidis dan Pappis, 1997).



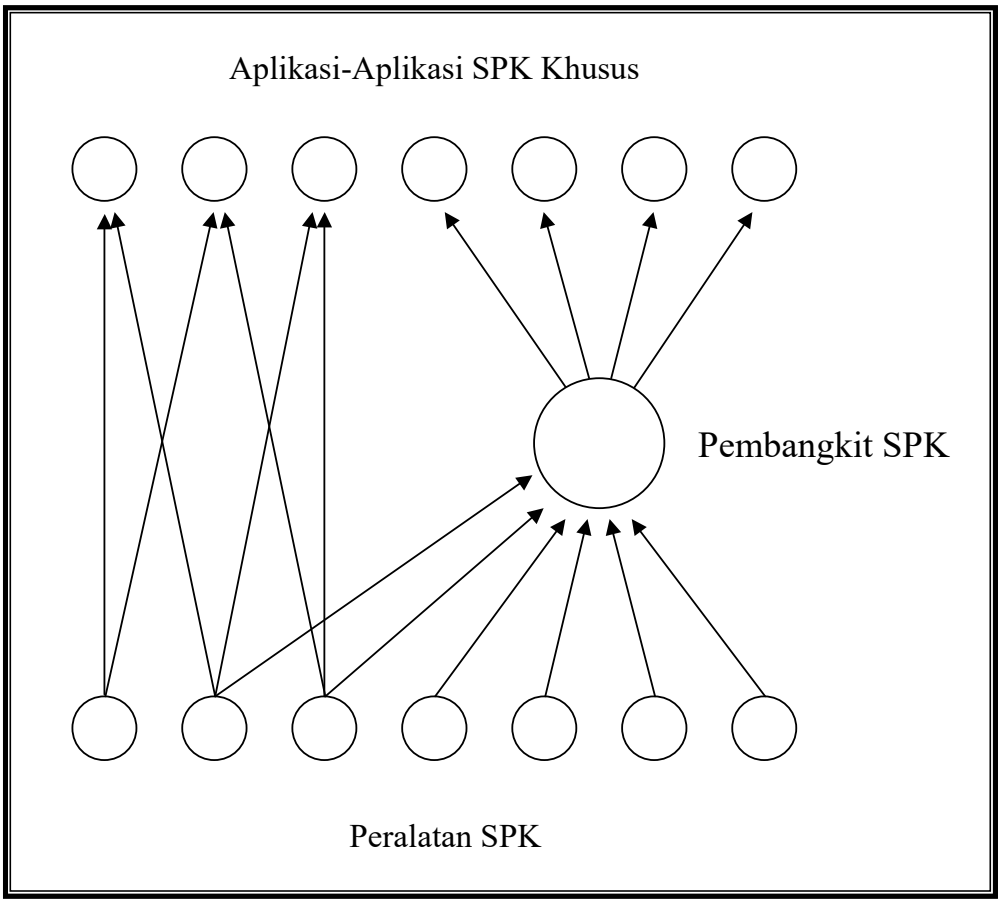
Gambar II.23 Subsistem Jaringan Komunikasi (Espinasse et. al., 1997)

Komponen jaringan komunikasi merupakan subsistem untuk memudahkan penetapan solusi suatu masalah dengan proses kerjasama antar pengambil keputusan sebagai tim. Proses ini perlu didukung oleh komunikasi elektronik. Rancangan jaringan kerja mengacu pada pengorganisasian distribusi hardware, software, dan data; dan bagaimana komponen tersebut dihubungkan dalam sistem (Power, 2000).

Proses yang terjadi dalam SPK dicirikan dengan hubungan umpan balik (*feedback*), artinya perilaku suatu elemen akan berpengaruh terhadap elemen lain, baik secara langsung maupun tidak langsung. Suatu sistem dikatakan cukup baik bilamana sanggup mempertahankan kondisi keseimbangannya terhadap perubahan lingkungan, atau dengan kata lain elemen-elemen yang dianggap berpengaruh (parameter) tidak boleh diabaikan dalam membangun sistem (Robert dan Michael, 1991).

6.2 Tingkat Teknologi SPK

Pembangunan SPK berdasarkan perangkatnya, mencakup tiga tingkat perangkat keras/lunak. Ketiganya digunakan berdasarkan perbedaan kemampuan teknik, dan perbedaan tugas yang akan dikerjakan, seperti diuraikan berikut ini.



Gambar II.24 Tingkat Teknologi SPK (Sprague dan Carlson, 1982)

SPK Khusus (*Spesific DSS*)

SPK yang siap digunakan secara langsung untuk menyelesaikan pekerjaan disebut SPK khusus. Sistem ini meliputi sistem informasi terapan, tetapi dengan karakteristik yang sangat berbeda dengan pemrosesan data biasa. SPK khusus adalah perangkat keras/lunak yang memungkinkan pembuat keputusan menyelesaikan sekumpulan masalah yang saling berhubungan. Contoh dari

SPK khusus ini adalah sistem interaktif grafik dalam evaluasi penjadualan produksi (Suryadi, 1992).

Pembangkit SPK (*DSS Generator*)

Pembangkit SPK adalah suatu paket perangkat keras dan lunak yang mempunyai kemampuan untuk mengembangkan SPK khusus secara cepat dan mudah. *Geodata Analysis and Display System* dari IBM, dan *Interactive Financial Planning System* dari *Executive Systems* merupakan contoh suatu pembangkit SPK (Sprague dan Carlson, 1982). SPK pembangkit diantaranya meliputi fasilitas penyiapan laporan, bahasa simulasi, tampilan grafik, subrutin statistik, dan sebagainya.

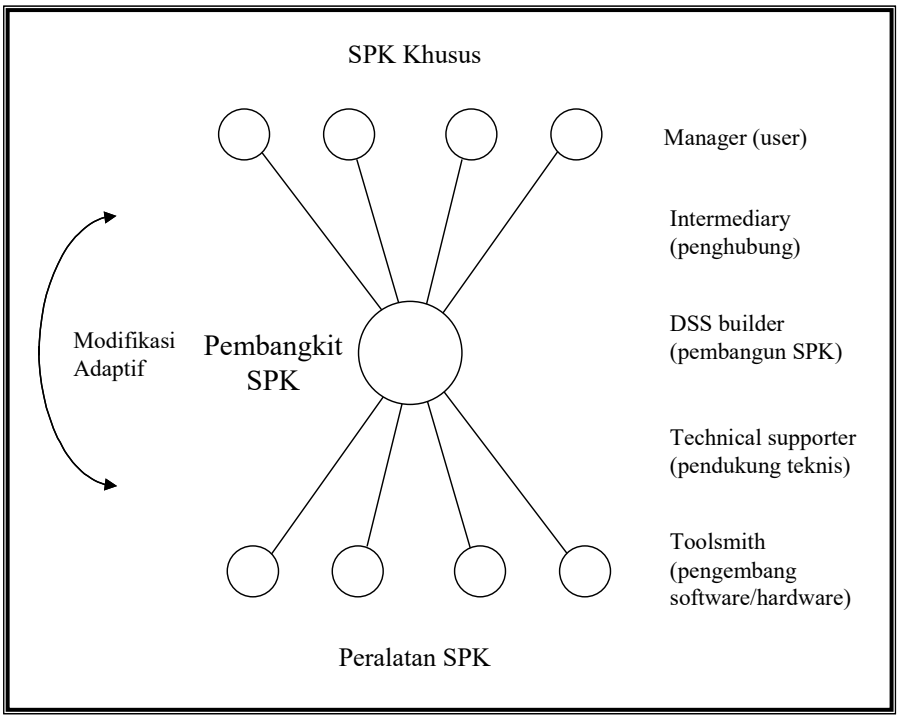
Peralatan SPK (*DSS Tools*)

Merupakan tingkatan teknologi yang paling mendasar dalam pengembangan SPK. Peralatan SPK adalah elemen-elemen perangkat keras dan lunak yang digunakan untuk mengembangkan SPK spesifik maupun pembangkit SPK. Yang termasuk dalam kategori-kategori teknologi ini antara lain bahasa-bahasa pemrograman (BASIC, FORTRAN, DBASE IV, C, PASCAL dan sebagainya), sistem operasi komputer khusus, perangkat lunak pengakses data, dan sebagainya.

Pemilihan peralatan SPK diukur berdasarkan tingkat efektivitas dan efisiensi peralatan tersebut dalam mendukung pengambil keputusan dalam melakukan pengambilan keputusan.

6.3 Pihak-pihak yang Berperan dalam Pengembangan SPK

Terdapat lima pihak yang berperan dalam pengembangan ketiga tingkatan SPK. Keterkaitan peran-peran dengan tingkat teknologi SPK disebut kerangka kerja pengembangan SPK.



Gambar II.25 Lima Pihak yang Berperan dalam Pengembangan SPK (Sprague dan Carlson, 1982)

Kelima peran tersebut adalah:

a *Manajer* atau *pemakai*, yaitu pihak yang terlibat langsung dengan proses pengambilan keputusan, pihak yang harus mengambil tindakan dan bertanggung jawab terhadap hasil tindakannya.

b *Penghubung*, yaitu pihak yang membantu pemakai, mungkin seorang asisten yang bertugas menjalankan terminal, atau lebih dari sekedar itu.

c *Pembangun SPK* atau *fasilitator*, yaitu pihak yang mengembangkan SPK khusus dari pembangkit SPK.

d *Pendukung teknik*, yaitu pihak yang mengembangkan tambahan kemampuan atau komponen sistem informasi yang dibutuhkan dalam pengembangan pembangkit SPK. Basis-basis data baru, model-model analisis baru, dan tambahan format tampilan data merupakan hasil kerja pendukung teknik.

e *Pengembang peralatan*, yaitu pihak yang mengembangkan teknologi baru (baik Hardware maupun Software), dan yang meningkatkan efisiensi hubungan antar subsistem dalam SPK.

6.4 Pendekatan Pengembangan dan Perancangan SPK

Kerangka teori dari penelitian tentang pengembangan dan perancangan SPK dikembangkan dalam elemen kunci, yang terdiri dari batasan hasil, konstruksi, dan hubungan antara konstruksi. Penelitian tersebut telah mengidentifikasi konstruksi SPK, yang terdiri dari delapan elemen, yaitu lingkungan, tugas, strategi implementasi, kapabilitas SPK, konfigurasi SPK, pengguna, perilaku pengguna, dan kinerja (Elerman et. al., 1995).

Ciri alamiah yang membedakan SPK dari sistem pemrosesan tradisional adalah teknik perancangannya. Karena tidak ada teori pengambilan keputusan yang komprehensif, dan karena kondisi yang dihadapi pengambil keputusan sering berubah secara cepat, pendekatan tradisional tidak lagi memadai. Pengembangan SPK membutuhkan pendekatan yang unik. Terdapat tiga jenis pendekatan dalam pengembangan SPK, dengan uraian sebagai berikut:

6.4.1 Analisis Sistem

Kebanyakan dari peralatan dan pendekatan untuk analisis sistem didasarkan pada asumsi bahwa sistem komputer akan mempunyai proses yang terdefinisi (seperti diagram alir). Karena sifat aslinya, SPK membutuhkan proses yang bebas dari asumsi diatas. Salah satu bentuk pendekatan proses bebas adalah pendekatan ROMC (*Representations, Operations, Memory Aids, Control Mechanisms*).

Pendekatan ROMC ini didasarkan pada representasi yang digunakan oleh pengambil keputusan untuk membuat konsep masalah, operasi terhadap representasi di atas, bantuan memori dan mekanisme kontrol. Adapun komponen ROMC tersebut ialah:

a Representasi

Berbagai aktivitas dalam proses pengambilan keputusan terjadi dalam konteks konseptualisasi informasi yang digunakan dalam aktivitas tersebut. Konseptualisasi ini mungkin berupa peta, sebuah gambar, sebuah grafik, beberapa angka, sebuah persamaan, dan sebagainya. Konseptualisasi ini bisa merupakan usaha mental, tetapi dalam banyak kasus merupakan usaha fisik menulis atau menggambar dalam kertas tulis, papan tulis, kertas grafik, dan sebagainya. Representasi fisik ini akan menjadi penting apabila pengambil keputusan hendak mengkomunikasikan beberapa aspek keputusan kepada pihak lain.

b Operasi

Dalam kaitannya dengan proses pengambilan keputusan, hal ini menyangkut kemampuan SPK dalam melakukan operasi analisis dan manipulasi terhadap representasi-representasi diatas.

c Bantuan Memori

Beberapa jenis bantuan memori dapat disediakan dalam sebuah SPK untuk mendukung penggunaan representasi dan operasi, diantaranya ialah:

- Basis data dari sumber data internal dan eksternal
- Kapasitas basis data, yaitu bantuan memori yang berisi spesifikasi pembagian atau pengelompokkan data, agregasi, dan sub kumpulan data.
- Tempat kerja (*workspace*) untuk menampilkan representasi dan melindungi hasil lanjutan yang dihasilkan oleh operasi
- Penghubung guna mengkaitkan data dari sebuah tempat kerja atau perpustakaan yang dibutuhkan sebagai referensi ketika dioperasikan dalam tempat kerja yang lain.
- Pemacu untuk mengingatkan seorang pengambil keputusan, bahwa beberapa operasi perlu segera dilaksanakan.
- Profil untuk menyimpan data status dan data standar (*default*).

d Mekanisme Kontrol

Representasi, operasi dan bantuan memori sebuah SPK dimaksudkan untuk mendukung berbagai variasi proses pengambilan keputusan dan variasi jenis-jenis keputusan. Bantuan kontrol SPK dimaksudkan untuk membantu pengambil keputusan menggunakan representasi, operasi dan memori dalam melaksanakan proses pengambilan keputusan berdasarkan gaya individual, keterampilan, dan pengetahuan yang dimilikinya. Terdapat beberapa bantuan kontrol. Jenis pertama terdiri dari bantuan yang memudahkan penggunaan SPK. Contohnya menu atau tombol fungsi untuk operasi pemilihan. Jenis kedua meliputi bantuan untuk mendukung latihan dan memberikan penjelasan tentang penggunaan SPK. Hal ini membantu pengambil keputusan belajar mengontrol SPK. Pesan-pesan kesalahan, perintah bantu, dan metode latihan yang memungkinkan pemakai belajar sambil bekerja, merupakan contoh dari bantuan jenis ini. Kontrol SPK dapat juga berupa pemanduan operasi yang relevan dengan satu atau beberapa representasi ke dalam prosedur. Jenis lain dari bantuan kontrol terdiri dari operasi pengambilan keputusan untuk mengubah hasil operasi lain, seperti kemampuan untuk mengedit hasil sebuah model peramalan. Akhirnya, bantuan kontrol dapat meliputi operasi untuk mengubah nilai-nilai standar (*default*) SPK. Sebagai contoh, jika SPK menyediakan operasi untuk membuat grafik dengan skala standar, operasi ini memungkinkan untuk mengubah skala tersebut.

6.4.2 Perancangan Iteratif

Rancangan SPK harus memungkinkan untuk mengubahnya secara cepat dan mudah. Untuk itu keempat tahap dalam proses pengembangan sistem biasa (analisis, perancangan, pembangunan dan penerapan) dikombinasikan ke dalam satu tahap tetapi dengan perulangan (Sprague dan Carlson, 1982). Partisipasi pihak pemakai sangat berperan dalam proses perancangan dengan metoda ini.

6.4.3 Sistem Adaptif

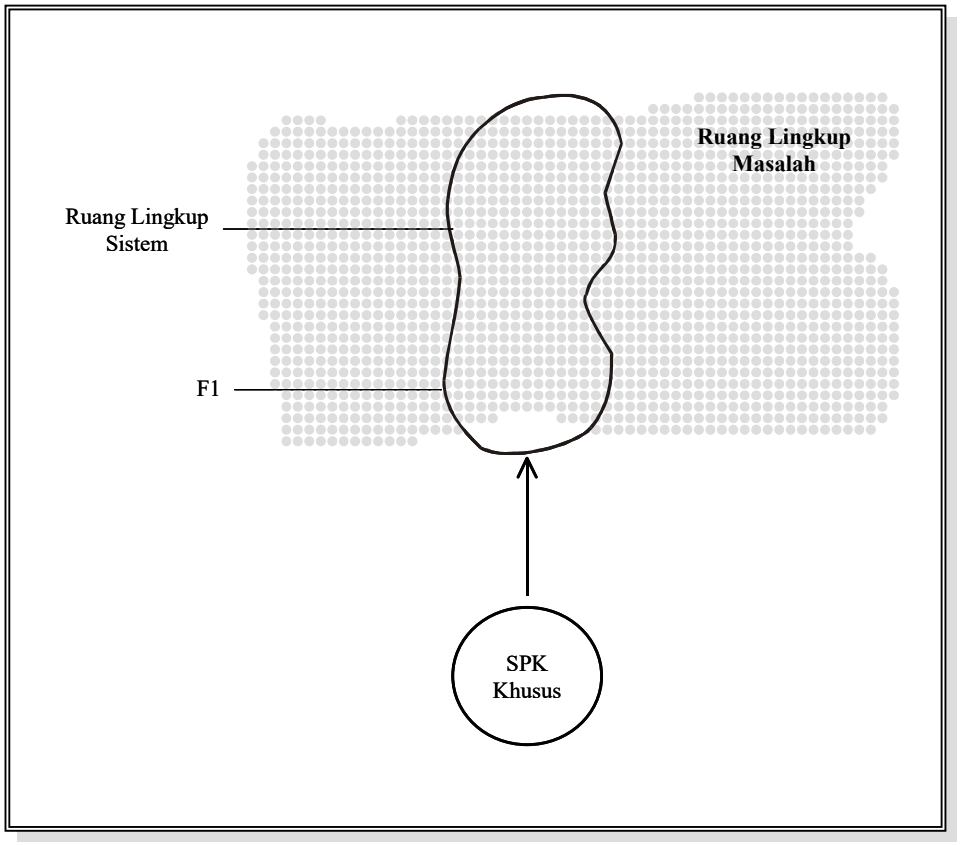
Dalam pengertian yang lebih luas, SPK adalah sistem adaptif yang terdiri dari ketiga tingkatan teknologi, dioperasikan oleh semua peran, dengan teknologi yang disesuaikan dengan perubahan sepanjang waktu.

Komponen dialog, basis data, dan pemodelan memberikan kemampuan representasi, operasi, memori dan kontrol yang diperlukan agar semua SPK berjalan efektif. Pendekatan ini menekankan pada perancangan komponen SPK yang dapat dikonfigurasi ulang setiap kali ada perubahan pada lingkungan yang ada.

6.4.4 Fleksibilitas dalam Pengembangan SPK

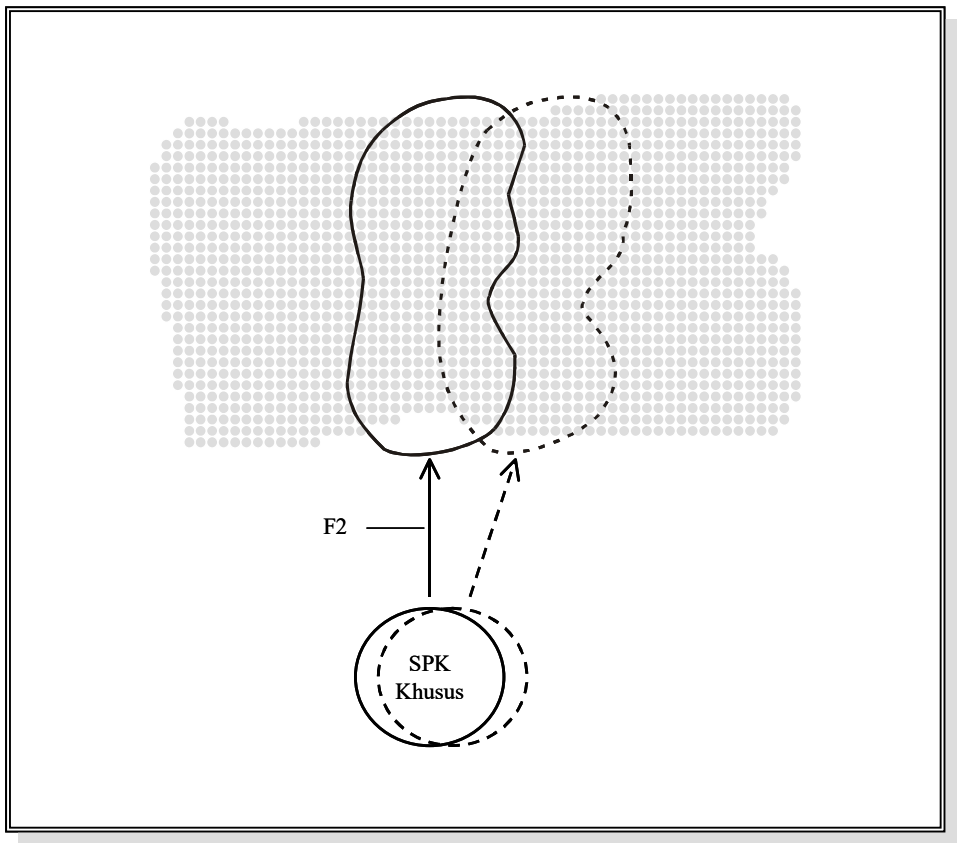
Salah satu sifat utama dari SPK yang dibangun adalah fleksibel, mengingat lingkungan, tugas-tugas dan pemakainya seringkali mengalami perubahan. Sprague dan Carlson (1982) membagi fleksibilitas SPK ke dalam empat tingkat, dengan penjelasan masing-masing tingkatan sebagai berikut:

1 ***Fleksibilitas tingkat pertama*** (F1), yaitu fleksibilitas yang memberikan kemampuan pada pemakai untuk menghadapi suatu masalah secara fleksibel dengan caranya sendiri. Fleksibilitas yang dimaksud ini adalah untuk menampilkan kemampuan pemahaman (*intelligence*), perancangan (*design*), dan pemilihan (*choice*) serta kemampuan untuk menggali beberapa alternatif untuk menyelesaikan suatu masalah. Pada Gambar II.26 terdapat lingkungan permasalahan (*problem space*), dimana setiap titiknya merupakan masalah atau sub masalah yang sedang dihadapi oleh pengambil keputusan. Dengan SPK yang mempunyai fleksibilitas F1, pengambil keputusan dapat membuat solusi dari masalahnya. Masalah yang dapat dipecahkan oleh SPK Khusus adalah masalah yang sudah dibatasi oleh sistem permasalahannya.



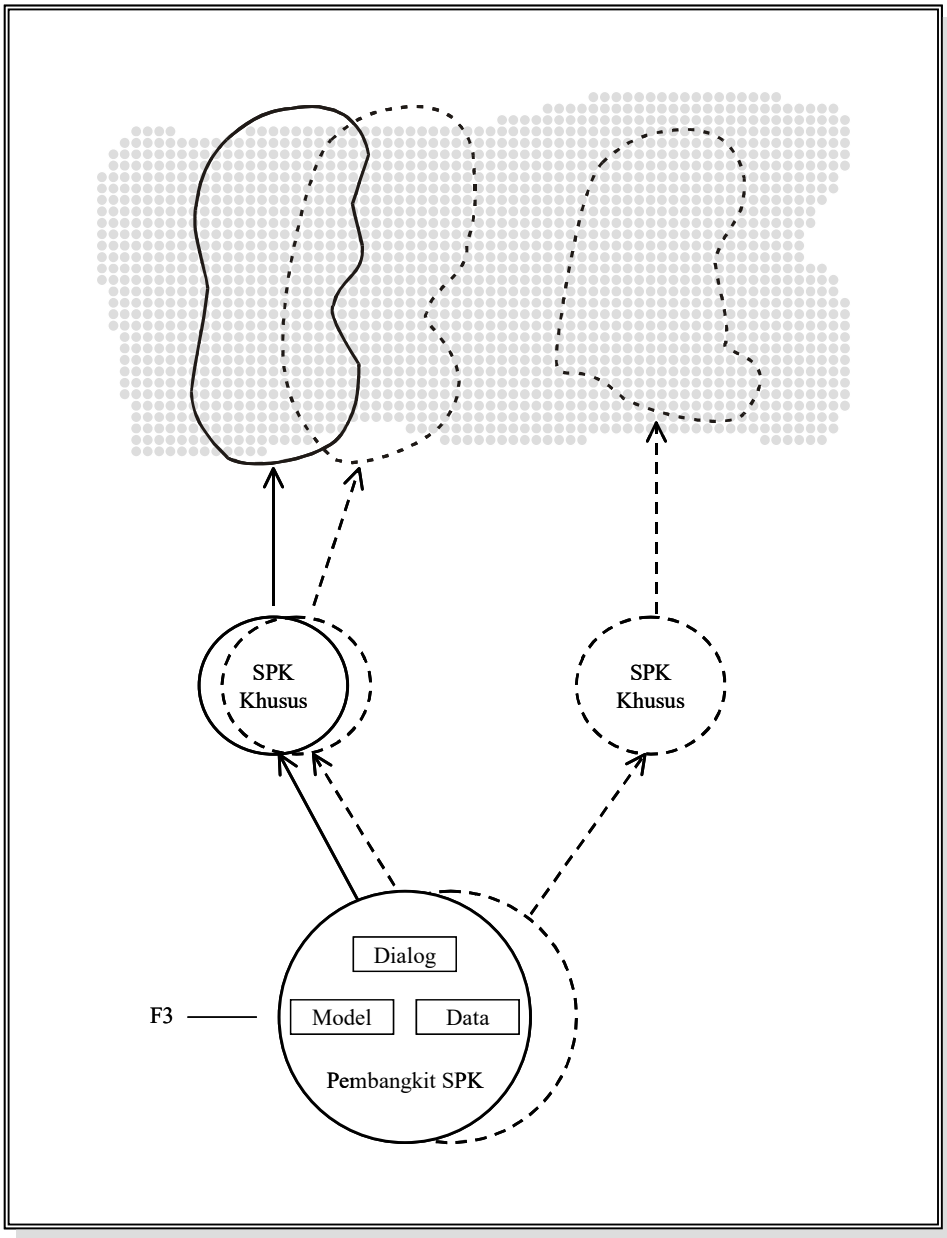
Gambar II.26 Fleksibilitas untuk Menyelesaikan Masalah - F1 (Sprague dan Carlson, 1982).

2 ***Fleksibilitas tingkat kedua*** (F2), merupakan fleksibilitas yang memiliki kemampuan untuk memodifikasi konfigurasi sebuah SPK Khusus sehingga dapat digunakan untuk menangani sekumpulan masalah yang berbeda atau diperluas (titik-titik di dalam ruang masalah). Fleksibilitas ini diimplementasikan dengan penambahan dan penghapusan representasi, operasi, bantuan memori dan mekanisme kontrol. Fleksibilitas tingkat F2 merupakan kemampuan untuk menambah/menghapus sebuah grafik, atau peta, operasi pada sebuah grafik, atau pada menu pilihan. Mengingat SPK Khusus adalah merupakan sekumpulan ROMC yang unik, maka fleksibilitas F2 dapat didefinisikan sebagai SPK Khusus yang berbeda.



Gambar II.27 Fleksibilitas SPK Khusus untuk Modifikasi - F2 (Sprague dan Carlson, 1982).

3 ***Fleksibilitas tingkat ketiga*** (F3), adalah fleksibilitas yang memiliki kemampuan untuk melakukan adaptasi terhadap perubahan yang cukup luas sehingga membutuhkan SPK Khusus yang sama sekali berbeda. Pada hakekatnya fleksibilitas F3 diimplementasikan melalui perubahan pada SPK Pembangkit, sehingga dapat digunakan untuk membangun SPK Khusus yang sebelumnya tidak mungkin dibuat. Perubahan ini dilakukan dengan memberi kemampuan baru dalam dialog, manajemen data dan pemodelannya, yang merupakan tambahan dari fasilitas pembangkit yang sudah ada.



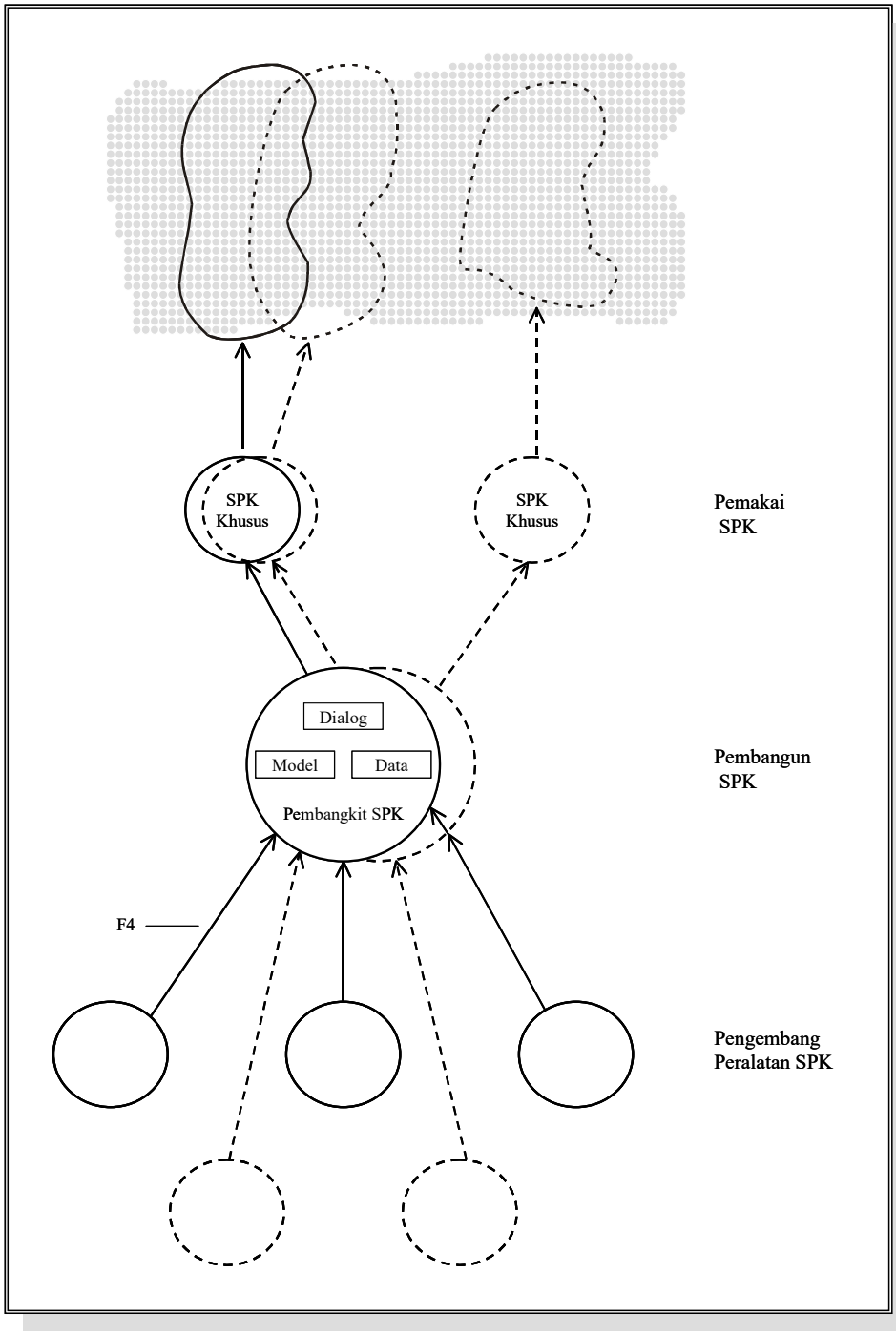
Gambar II.28 Fleksibilitas SPK Khusus untuk Adaptasi - F3 (Sprague dan Carlson, 1982).

Terdapat dua jenis penambahan atau penghapusan Dialog-Data-Model ini, yaitu sebagai berikut:

a Penambahan atau penghapusan kemampuan yang dimiliki oleh SPK Pembangkit. Misalnya, penambahan atau penghapusan menu, sumber data, pilihan analisis pemodelan.

b Penambahan atau penghapusan kemampuan dari atau menuju pembangkit. Contohnya adalah kemampuan untuk menambah atau menghapus gaya dialog (panel sentuh), struktur data logik (hirarki atau jaringan kerja) yang mampu untuk menambah atau menghapus pendekatan pemodelan atau analisis (Simulasi Monte Carlo). Dengan demikian, terjadi perluasan dalam batasan masalahnya.

4 ***Fleksibilitas tingkat keempat*** (F4), yaitu fleksibilitas kemampuan sistem untuk menjawab perubahan yang terjadi secara alamiah dari teknologi yang mendasari perancangan SPK. Hal ini dapat diimplementasikan dengan terjadinya perubahan pada peralatan dan kemampuan teknis, yang dapat meningkatkan kekuatan pembangit untuk ber-adaptasi dengan perkembangan teknologi. Fleksibilitas F4 juga melengkapi kemampuan SPK Pembangkit untuk menyerap teknologi guna memperbaiki kemampuan adaptasinya dalam menangani berbagai masalah yang dihadapi oleh pemakai. Hal ini mungkin berupa pengembangan teknologi perangkat keras atau perangkat lunak. Peralatan dan teknologi baru ini ditandai dengan garis putus-putus yang terdapat pada Gambar II.29.



Gambar II.29 Fleksibilitas SPK Khusus untuk Berkembang - F4 (Sprague dan Carlson, 1982).

BAB III
METODOLOGI PENELITIAN

1 Sistematika Penelitian

Sistematika pemecahan masalah dimulai dari tinjauan ide dasar perkembangan model pengambilan keputusan, khususnya AHP. Analisa ini kemudian dilanjutkan dengan pengembangan model berdasarkan tinjauan beberapa elemen-elemen keputusan, yang memiliki kesesuaian dengan masalah pada dunia nyata, yang bertujuan untuk optimalisasi utilitas pengambil keputusan melalui pendekatan model pengambilan keputusan kriteria majemuk.

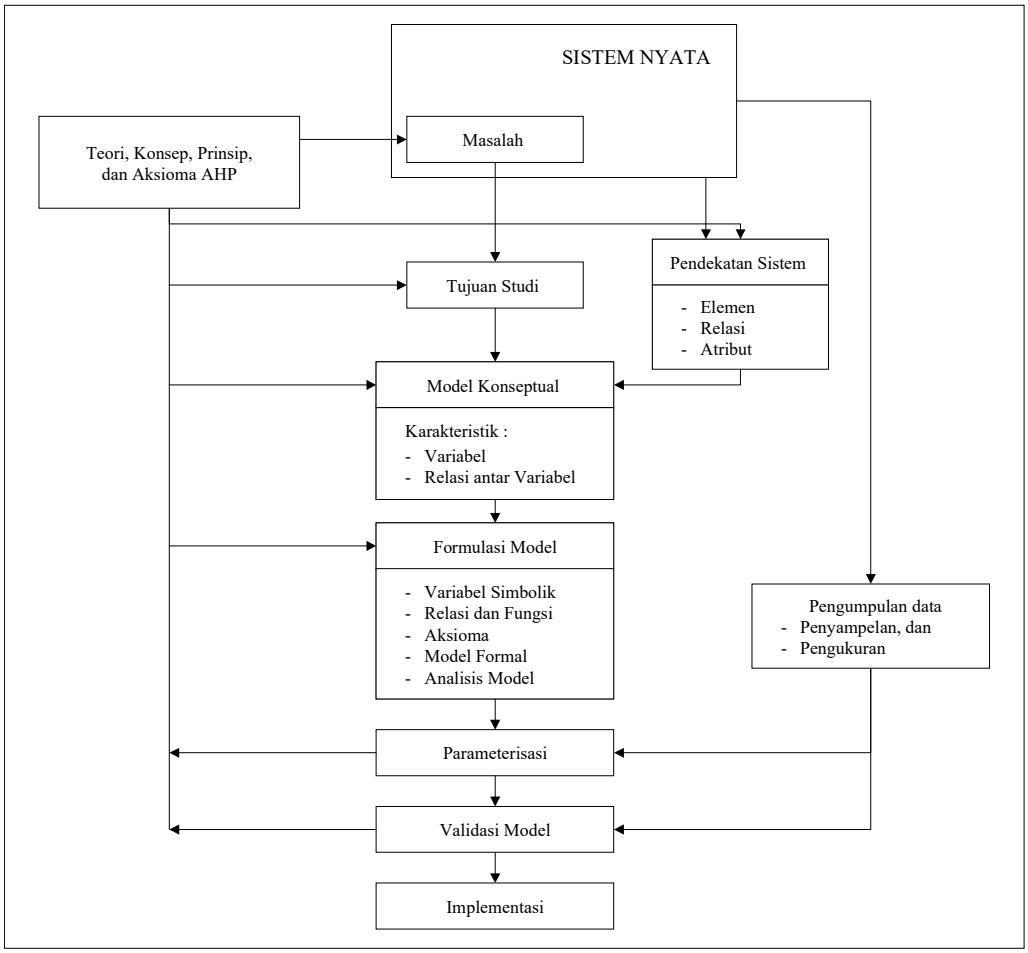
Tahapan-tahapan yang dilakukan pada penelitian ini pada garis besarnya terdiri dari:

1 Penentuan tujuan penelitian
2 Studi pendahuluan;
3 Perancangan model pengambilan keputusan, yang terdiri dari:
   a definisi masalah,
   b perumusan model konseptual,
   c formulasi model,
   d validasi model;
4 Perancangan sistem pendukung keputusan kriteria majemuk, yang terdiri dari:
   a analisis sistem,
   b desain sistem,
   c konstruksi sistem

2. Pengembangan Model

Pengembangan model dalam penelitian ini dilakukan untuk memperoleh model yang relatif baru, sebagai pengembangan dari AHP. Artinya, pengembangan model dilakukan dengan meninjau model AHP, kemudian dilakukan

modifikasi, kombinasi, dan atau penambahan komponen model yang baru. Model ini kemudian diformalkan, dan diuji kesesuaiannya dengan sistem nyata secara ilmiah. Sistematika perancangan model dalam penelitian ini mengikuti siklus pemodelan seperti pada Gambar III.1.



Gambar III.1 Langkah-Langkah Pengembangan Model

Sebagai langkah awal dalam pengembangan model dilakukan definisi masalah. Selanjutnya berdasarkan definisi masalah dirumuskan model konseptual yang menunjukkan keterkaitan antara variabel yang menentukan perilaku model. Model ini termasuk model verbal yang hanya menguraikan hubungan masalah, sistem, dan tujuan studi. Tujuan studi memberikan indikasi performansi yang ingin dicapai dan model konseptual memberikan kerangka yang membentuk

performansi yang diharapkan. Untuk mengoperasionalkan model konseptual simbolisasi dan penetapan aturan kuantitatif. Idealisasi dan penyederhanaan keterkaitan variabel model disebut sebagai tahap karakterisasi model.

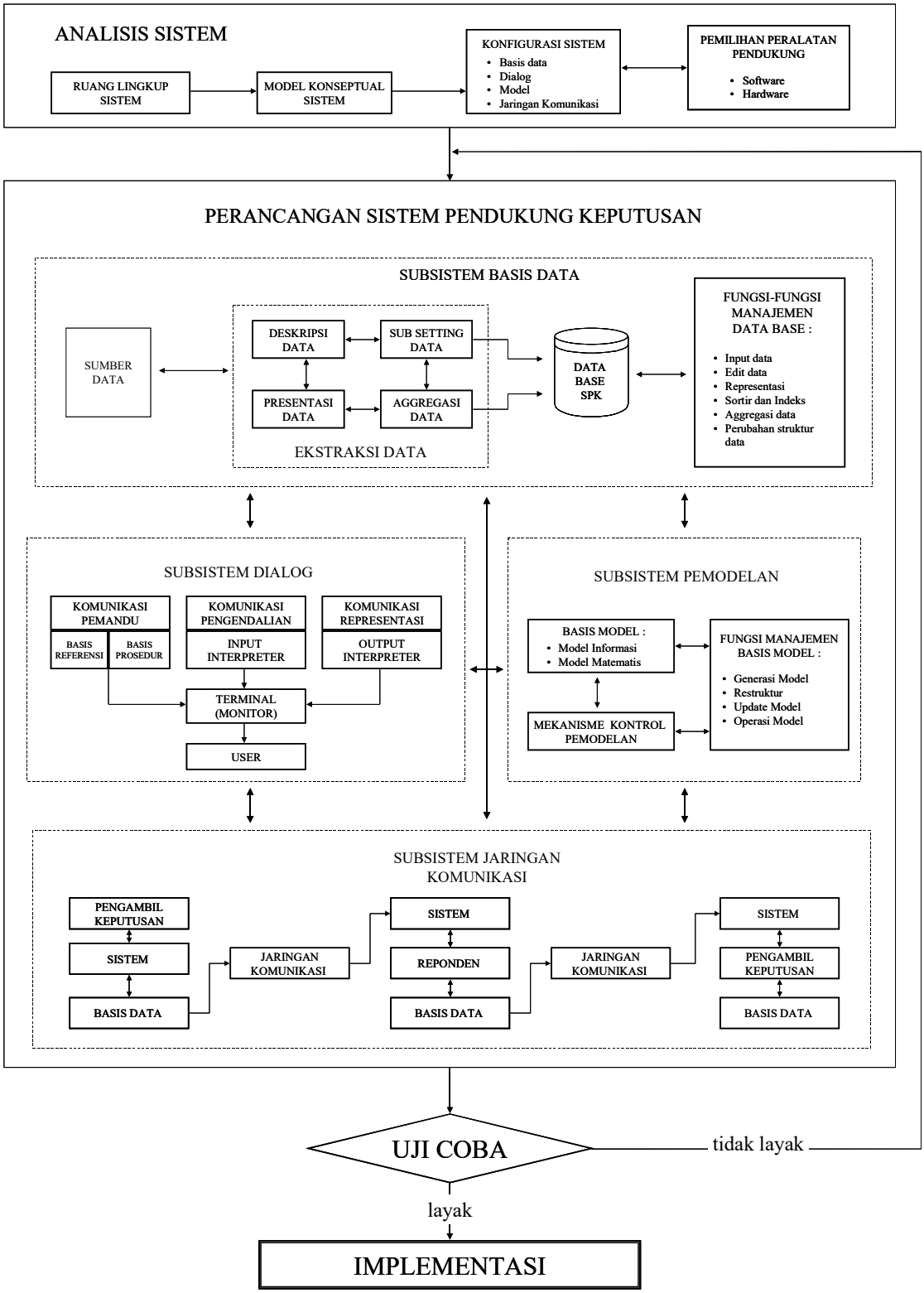
Formulasi model dilakukan sebagai awal pengembangan model formal yang menunjukkan ukuran performansi model sebagai fungsi dari variabel-variabel model. Dalam formulasi model digunakan prinsip teleologik (meninjau tujuan pemodelan) untuk memfungsionalkan atribut-atribut dengan melihat tujuan (*teleos*) dari sistem. Melalui pendekatan sistem, eksistensi sistem dan lingkungannya dapat dipahami dengan diketahuinya elemen-elemen sistem, relasi antar elemen, dan atribut dari masing-masing elemen. Lingkungan sistem merupakan kumpulan objek di luar batasan (*boundaries*) sistem yang mempengaruhi (dipengaruhi) sistem.

Setelah formulasi awal model selesai, maka kemampuan model untuk mereproduksi sifat-sifat dan perilaku sistem nyata dilakukan pengujian. Dalam hal ini dilakukan pengujian berdasarkan tiga kriteria untuk mengevaluasi sebuah model, yaitu:

a Ketelitian, diuji kesesuaian perilaku model dengan perilaku sistem nyata yang direpresentasikannya.
b Validitas, dilakukan pengujian struktur antar variabel model.
c Ketersediaan taksiran untuk variabel, dilakukan pengujian ketersediaan nilai taksiran untuk variabel-variabel kunci.

3 Perancangan SPKKM

Perancangan SPKKM dilakukan dengan melakukan penelitian terhadap kebutuhan-kebutuhan sistem. Perancangan SPKKM pada penelitian ini mengikuti bagan alir pada Gambar III.2.



Gambar III.2 Sistematika Perancangan SPKKM

3.1 Analisis Sistem

Analisis sistem pada penelitian ini merupakan studi perancangan pendahuluan, yang dilakukan guna merumuskan kerangka dan ruang lingkup SPKKM, persyaratan unjuk kerja SPKKM, memilih konsep-konsep, analisa dan aplikasi model pembuatan keputusan yang relevan dengan tujuan SPKKM yang akan dibangun, dan identifikasi spesifikasi SPKKM.

3.2 Perancangan Sistem

Kegiatan yang dilakukan dalam proses perancangan sistem diawali dengan merumuskan spesifikasi SPKKM, kemudian dilanjutkan dengan perancangan konfigurasi SPKKM.

Perancangan konfigurasi sistem dilakukan untuk melihat sistem informasi yang diperlukan SPKKM, yang meliputi aliran data, pengolahan data, dilanjutkan dengan pemilihan komponen perangkat keras dan perangkat lunak yang akan dipakai dalam pengembangan SPKKM. Proses selanjutnya dalam merancang komponen teknologi SPKKM (basis data, model keputusan, model dialog, dan jaringan komunikasi) sesuai kebutuhan performansi sistem usulan.

Adapun pendekatan yang diterapkan dalam perancangan komponen-komponen teknologi SPKKM dititikberatkan pada pertimbangan kemampuan tiap komponen tersebut dalam hal *Representasi, Operasi, Memori* dan *Mekanisme Kontrol* (pendekatan ROMC), untuk kemudian memadukannya menjadi suatu sistem. Dengan demikian, SPKKM hasil perancangan akan mudah dikembangkan lebih lanjut atau disesuaikan dengan perubahan-perubahan dalam situasi operasinya.

BAB IV
PENGEMBANGAN MODEL

Bab ini menyajikan kajian tentang pengembangan model, mengingat pengembangan model berbasis AHP, maka sistematika penyajiannya diawali dengan deskripsi tentang AHP dan beberapa topik yang berkaitan dengan fokus penelitian, kemudian dilanjutkan dengan deskripsi pengembangan model.

1 *Analityc Hierarchy Process*

Pendekatan yang dilakukan dalam AHP adalah analisis permasalahan komplek melalui prinsip-prinsip **dekomposisi**, **analisis perbandingan**, dan **sintesa prioritas**.

Secara umum langkah-langkah penyelesaian menggunakan AHP dalam pemecahan masalahnya, mengikuti langkah-langkah sebagai berikut:

1 Mendefinisikan masalah dan menentukan solusi yang diinginkan.

2 Membuat struktur hirarki yang diawali dengan tujuan umum, dilanjutkan dengan kriteria, subkriteria, dan kemungkinan alternatif-alternatif pada tingkatan yang paling bawah.

3 Membuat matriks perbandingan berpasangan yang menggambarkan kontribusi relatif atau pengaruh setiap elemen terhadap masing-masing tujuan atau kriteria yang setingkat di atasnya. Perbandingan dilakukan berdasarkan penilaian dari pengambil keputusan dengan menilai tingkat kepentingan suatu elemen dibandingkan terhadap elemen lainnya.

4 Melakukan perbandingan berpasangan sehingga diperoleh penilaian seluruhnya sebanyak nx[(n-1)/2] buah, dengan n adalah banyaknya elemen yang dibandingkan.

5 Menghitung *eigenvalue* dan menguji konsistensinya, jika tidak konsisten maka pengambilan data diulangi.

6 Mengulangi langkah 3, 4, dan 5 untuk seluruh tingkat hirarki.

7 Menghitung *eigenvector* dari setiap matriks perbandingan berpasangan. Nilai *eigenvector* merupakan bobot setiap elemen. Langkah ini untuk mensintesis penilaian dalam penentuan prioritas elemen-elemen pada tingkat hirarki terendah sampai pencapaian tujuan.

8 Memeriksa konsistensi hirarki. Jika nilainya lebih dari 10 persen maka penilaian data penilaian harus diperbaiki.

1.1 Aksioma-Aksioma AHP

Pengertian aksioma, adalah sesuatu yang tidak dapat dibantah kebenarannya atau yang harus terjadi. Ada empat aksioma yang harus diperhatikan dalam penggunaan model AHP dan pelanggaran setiap aksioma berakibat tidak validnya model yang dipakai. Keempat aksioma tersebut adalah:

Aksioma 1:

Reciprocal, artinya pengambil keputusan harus dapat membuat perbandingan dan menyatakan preferensinya. Preferensi itu sendiri harus memenuhi syarat resiprokal yaitu kalau a_1 lebih disukai dari a_2 dengan skala x, maka a_2 lebih disukai dari a_1 dengan skala $1/x$.

Aksioma 2:

Homogeneity, artinya preferensi seseorang harus dapat dinyatakan dalam skala terbatas atau dengan kata lain elemen-elemennya dapat dibandingkan satu sama lain. Kalau aksioma ini tidak dipenuhi, maka elemen-elemen yang dibandingkan tersebut tidak homogenous atau harus dibentuk suatu *cluster* (kelompok elemen-elemen) yang baru.

Aksioma 3:

Dependence, artinya preferensi dinyatakan dengan mengasumsikan bahwa kriteria tidak dipengaruhi oleh alternatif-alternatif melainkan oleh tujuan secara keseluruhan. Hal ini menunjukkan ketergantungan atau pengaruh dalam model

AHP adalah searah ke atas. Artinya, perbandingan antar elemen dalam satu level dipengaruhi atau tergantung pada elemen-elemen dalam level di atasnya.

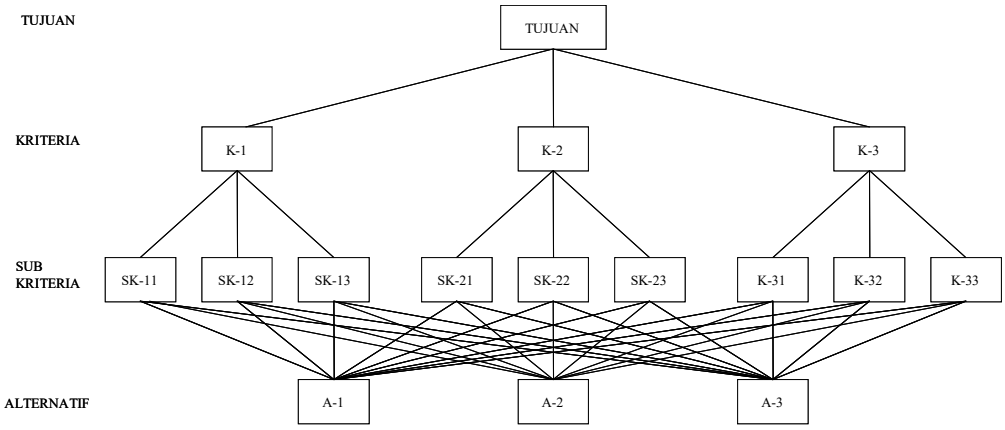
Aksioma 4:

Expectations, untuk tujuan pengambilan keputusan, struktur hirarki diasumsikan lengkap. Apabila asumsi ini tidak dipenuhi, maka pengambil keputusan dapat dikatakan tidak memakai seluruh kriteria sehingga keputusan yang diambil dianggap tidak lengkap.

1.2 Hirarki Tujuan

Hirarki adalah alat yang paling mudah untuk memahami masalah yang kompleks dimana masalah tersebut diuraikan ke dalam elemen-elemen yang bersangkutan, menyusun elemen-elemen tersebut secara hirarkis, dan akhirnya melakukan penilaian atas elemen-elemen tersebut, sekaligus menentukan keputusan yang akan diambil.

Hirarki yang digunakan dalam proses pengambilan keputusan dalam model AHP adalah bentuk hirarki fungsional yang menguraikan masalah yang kompleks menjadi bagian-bagian sesuai dengan hubungan esensialnya. Pembentukkan hirarki pada prinsipnya adalah suatu tujuan yang bersifat umum dijabarkan dalam beberapa sub tujuan yang lebih terperinci, yang dapat menjelaskan yang dimaksud dalam tujuan pertama. Penjabaran ini dapat dilakukan terus hingga akhirnya diperoleh tujuan yang bersifat operasional. Dan pada tingkat hirarki inilah dilakukan proses evaluasi atas alternatif-alternatif, yang merupakan ukuran dari pencapaian tujuan utama. Pada hirarki ini dapat ditetapkan dalam satuan apa kriteria diukur, sehingga setiap alternatif dapat diukur secara operasional.



Gambar IV.1 Model Hirarki Tujuan

Penjabaran tujuan dalam hirarki yang lebih rendah pada dasarnya ditujukan agar memperoleh kriteria yang dapat diukur. Walaupun sebenarnya tidaklah selalu demikian keadaannya. Dalam beberapa hal tertentu, mungkin lebih menguntungkan bila menggunakan tujuan pada hirarki yang lebih tinggi dalam proses analisis. Semakin rendah dalam menjabarkan suatu tujuan, semakin mudah pula penentuan ukuran objektif dari kriteria-kriterianya. Tetapi ada kalanya dalam proses analisis pengambilan keputusan tidak memerlukan penjabaran yang terlalu terperinci. Bila demikian keadaannya, maka salah satu cara untuk menyatakan ukuran pencapaiannya adalah dengan menggunakan skala subjektif.

Untuk memastikan bahwa kriteria-kriteria yang dibentuk sesuai dengan tujuan permasalahan, maka perlu dilihat sifat-sifat berikut:

1 Minimum

Jumlah kriteria diusahakan tidak terlalu banyak dan berlebihan untuk memudahkan analisis.

2 Independen

Setiap kriteria tidak saling bergantung/tumpang tindih dan harus dihindarkan pengulangan kriteria untuk suatu maksud yang sama.

3 Lengkap

Kriteria harus dapat mencakup seluruh aspek penting dalam persoalan.

4 Operasional

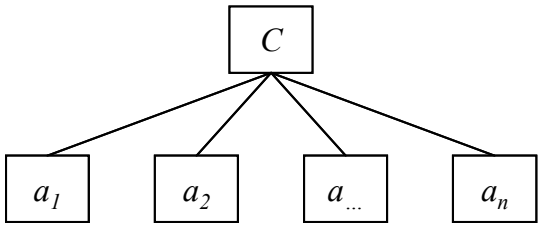
Kriteria harus dapat diukur dan dianalisa, baik secara kuantitatif maupun kualitatif, dan dapat dikomunikasikan.

Berdasarkan atas suatu penelitian Psikologi yang dilakukan oleh G.A. Miller, pada tahun 1965 yang menyimpulkan bahwa manusia tidak dapat secara simultan membandingkan lebih dari tujuh objek (tambah atau kurang dua). Pada kondisi tersebut, manusia akan mulai kehilangan konsistensinya dalam melakukan perbandingan dan bahkan cenderung menjadi bingung. Untuk manusia yang tergolong luar biasa, paling banyak ia dapat melakukan perbandingan sembilan elemen secara konsisten. Lebih dari itu, hampir tidak mungkin. Sedangkan orang biasa kebanyakan mampu membandingkan paling sedikit lima elemen secara konsisten (Brodjonegoro, 1992). Hal ini mendasari pembuatan percabangan hirarki dalam AHP diupayakan tidak lebih dari tujuh elemen.

Pengambilan keputusan AHP memberikan bobot prioritas untuk sejumlah *n* alternatif dengan mempertimbangkan sejumlah *m* kriteria. Dalam hal ini, kriteria-kriteria dinyatakan sebagai C_i (untuk i = 1, 2, 3, ..., m) dan alternatif-alternatif sebagai a_i (untuk i = 1, 2, 3, ..., n).

1.3 Penyusunan Matriks Perbandingan

Misalkan terdapat subsistem hirarki dengan satu kriteria *C* dan sejumlah *n* elemen di bawahnya: a_1 sampai a_n, seperti terlihat pada Gambar IV.2.



Gambar IV.2 Subsistem Hirarki

Perbandingan antar elemen untuk subsistem hirarki itu dibuat dalam bentuk matriks *n*x*n*, seperti disajikan pada Gambar IV.3. Matriks tersebut dinamakan matriks perbandingan berpasangan.

C	a_1	a_2	...	a_n
a_1	...	...	...	...
a_2	...	...	...	...
. . .	. . .	. . .	. . .	. . .
a_n	...	...	...	...

Gambar IV.3 Matriks Perbandingan Berpasangan

1.4 Pengisian Matriks Perbandingan

Setiap elemen yang terdapat dalam hirarki harus diketahui bobot relatifnya satu sama lain. Tujuannya adalah untuk mengetahui tingkat kepentingan/preferensi para pengambil keputusan terhadap elemen dan struktur hirarki secara keseluruhan.

Langkah pertama dalam menentukan susunan prioritas elemen adalah dengan menyusun perbandingan berpasangan (*pairwise comparison*), yaitu membandingkan dalam bentuk berpasangan seluruh elemen untuk setiap subsistem hirarki. Perbandingan tersebut kemudian ditransformasikan dalam bentuk matriks untuk maksud analisis numerik.

Penilaian perbandingan antar elemen dalam hirarki tersebut menggunakan skala penilaian satu sampai sembilan, dengan perincian seperti tampak pada Tabel IV.1.

Tabel IV.1 Skala Penilaian Perbandingan

Intensitas kepentingan	Keterangan	Penjelasan
1	Kedua elemen sama pentingnya	Dua elemen mempunyai pengaruh yang sama besar terhadap tujuan
3	Elemen yang satu sedikit lebih penting dari pada elemen yang lainnya	Pengalaman dan penilaian sedikit menyokong satu elemen dibandingkan atas elemen lainnya
5	Elemen yang satu lebih penting dari pada elemen yang lainnya	Pengalaman dan penilaian sangat kuat menyokong satu elemen dibandingkan atas elemen lainnya
7	Satu elemen jelas lebih penting dari pada elemen lainnya	Satu elemen yang kuat disokong dan dominannya telah terlihat dalam praktek
9	Satu elemen mutlak penting dari pada elemen lainnya	Bukti yang mendukung elemen yang satu terhadap elemen lain memiliki tingkat penegasan tertinggi yang mungkin menguatkan
2, 4, 6, 8	Nilai-nilai antara dua nilai per-timbangan yang berdekatan	Nilai ini diberikan bila ada dua kompromi di antara dua pilihan
Kebalikan	Jika untuk elemen *i* mendapat satu angka bila dibandingkan dengan elemen *j*, maka *j* mempunyai nilai kebalikannya bila dibandingkan dengan *i*	

Pengambil keputusan harus memberikan penilaian sebanyak $n[(n\text{-}1)/2]$ untuk setiap matriks berukuran nxn.

1.5 Perhitungan Nilai Bobot

Hasil penilaian pengambilan keputusan disajikan pada matriks yang berisi nilai penilaian, seperti terlihat pada Gambar IV.4.

C	a_1	a_2	...	a_n
a_1	a_{11}	a_{12}	...	a_{1n}
a_2	a_{21}	a_{22}	...	a_{2n}
. . .	. . .	. . .	. . .	. . .
a_n	a_{n1}	a_{n2}	...	a_{nn}

Gambar IV.4 Matriks Nilai Perbandingan Berpasangan

Nilai a_{ij} adalah nilai perbandingan elemen a_i terhadap elemen a_j, yang menyatakan hubungan seberapa besar tingkat kepentingan elemen a_i bila dibandingkan dengan elemen a_j, atau seberapa besar elemen a_i disukai dibandingkan dengan elemen a_j terhadap kriteria C.

Bila diketahui nilai perbandingan a_i terhadap a_j adalah a_{ij} maka secara teoritis nilai perbandingan a_j terhadap a_i (*reciproc*) atau nilai a_{ji} adalah $1/a_{ij}$. Sedangkan nilai a_{ij} dalam situasi $i=j$ adalah mutlak sama dengan satu. Dengan demikian, bentuk matriks A adalah sebagai berikut:

$$A = \begin{bmatrix} 1 & a_{12} & \dots & a_{1n} \\ 1/a_{12} & 1 & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ 1/a_{1n} & 1/a_{2n} & \dots & 1 \end{bmatrix}$$

Bobot yang dicari dinyatakan dalam vektor w=(w_1, w_2, ..., w_n). Nilai w_n menyatakan bobot relatif elemen a_n terhadap seluruh himpunan elemen pada subsistem tersebut. Masalahnya adalah bagaimana mendapatkan bobot w_i untuk setiap nilai a_{ij} tersebut. Untuk memecahkan masalah tersebut, dapat dilakukan melalui tiga tahap berikut:

Tahap 1:

Diasumsikan bahwa nilai perbandingan merupakan hasil pengukuran nyata. Untuk membandingkan elemen a_1 dengan a_2 diambil patokan dari bobot setiap elemen. Dengan demikian, nilai perbandingan yang diperoleh dari partisipan berdasarkan penilaian Gambar IV.4, yaitu a_{ij} dapat dinyatakan dalam vektor w sebagai hubungan antara bobot w_i dengan hasil penilaian a_{ij} adalah sebagai berikut:

$$a_{ij} = \frac{w_i}{w_j} \quad ; i, j = 1, 2, \ldots, n \qquad \ldots(IV.1)$$

dan matriks perbandingannya adalah:

$$\begin{bmatrix} \frac{w_1}{w_1} & \frac{w_1}{w_2} & \cdots & \frac{w_1}{w_n} \\ \frac{w_2}{w_1} & \frac{w_2}{w_2} & \cdots & \frac{w_2}{w_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{w_n}{w_1} & \frac{w_n}{w_2} & \cdots & \frac{w_n}{w_n} \end{bmatrix}$$

Ternyata, bentuk hubungan di atas tidak realistik untuk menangani kasus yang sebenarnya (nyata). Pertama, karena pengukuran fisik tidak pernah eksak secara matematis sehingga diperlukan kelonggaran (*deviation*). Kedua, penyimpangan pada penilaian yang dilakukan manusia biasanya cukup besar.

Tahap 2:

Untuk melihat seberapa besar kelonggaran yang dibuat untuk penyimpangan, perhatikan baris ke-*i* dari matriks *A*. Elemen baris tersebut adalah:

$$a_{i1},\ a_{i2},\ \ldots,\ a_{in}$$

Pada kasus ideal (eksak), nilai-nilai ini sama dengan perbandingan:

$$\frac{w_i}{w_1},\quad \frac{w_i}{w_2},\quad \ldots,\quad \frac{w_i}{w_j},\quad \ldots,\quad \frac{w_i}{w_n}$$

Jika elemen pertama dari baris tersebut dikalikan dengan w_1, elemen kedua dengan w_2, dan seterusnya, akan diperoleh:

$$\left(\frac{w_i}{w_1}\right).w_1,\quad \left(\frac{w_i}{w_2}\right).w_2,\quad \ldots,\quad \left(\frac{w_i}{w_j}\right).w_j,\quad \ldots,\quad \left(\frac{w_i}{w_n}\right).w_n$$

hasilnya adalah baris dengan elemen yang identik: $w_i,\ w_i,\ \ldots,\ w_i,\ \ldots,\ w_i$.

Pada kasus umum, akan diperoleh elemen baris yang besarnya berkisar sekitar nilai w_i, sehingga beralasan jika dikatakan bahwa w_i adalah harga rata-rata dari nilai tersebut:

$w_i =$ rata-rata dari $((a_{i1}.w_1), (a_{i2}.w_2), \ldots, (a_{in}.w_n))$

$$w_i = \tfrac{1}{n}\sum_{j=1}^{n} a_{ij} w_j \ ; \qquad i = 1, 2, \ldots, n \qquad \ldots \text{(IV.2)}$$

Tahap 3:

Pada kasus nyata, nilai a_{ij} tidak selalu sama dengan w_i/w_j, sehingga akan mempengaruhi solusi persamaan (IV.2), kecuali jika n berubah. Untuk selanjutnya nilai n ini diganti oleh λ_{maks} sehingga:

$$w_i = \frac{1}{\lambda_{maks}} \sum_{j=1}^{n} a_{ij} w_j \; ; \quad i = 1, 2, \ldots, n \qquad \ldots \text{(IV.3)}$$

Persamaan (IV.3) memiliki solusi unik, yang dikenal dengan nilai *eigenvelue* (nilai eigen). Nilai λ_{maks} adalah *eigenvalue* maksimum dari matriks A.

Dari tahap 1, dapat diturunkan hubungan:

1 $a_{ij} \,.\, a_{jk} = (w_i / w_j) \,.\, (w_j / w_k)$

$a_{ij} \,.\, a_{jk} = (w_i / w_k)$

$a_{ij} \,.\, a_{jk} = a_{ik}$ untuk semua i, j, k … (IV.4)

Bentuk persamaan (IV.4) menyatakan harus terpenuhinya konsistensi penilaian dari elemen matriks tersebut.

2 $a_{ji} = (w_j / w_i)$

$a_{ji} = 1/(w_i / w_j)$

$a_{ji} = 1/a_{ij} \; ; \quad i, j = 1, 2, \ldots, n$ … (IV.5)

Bentuk persamaan (IV.5) menunjukkan ciri resiprokal dari matriks perbandingan.

Pada situasi penilaian yang konsisten sempurna (teoritis) maka didapatkan hubungan:

$a_{ik} = a_{ij}.a_{jk}$ untuk semua i, j, k

dan matriks yang didapatkan adalah matriks yang konsisten.

Dari persamaan di atas dapat dibuat persamaan berikut:

$$a_{ij} \cdot w_j / w_i = 1 \quad ; \quad i, j = 1, \ldots, n$$

dengan demikian didapatkan:

$$\sum_{j=1}^{n} a_{ij}.w_j \; 1/w_i = n \quad ; \quad i = 1, \ldots, n$$

$$\sum_{j=1}^{n} a_{ij}. w_j = n \, w_i \quad ; \quad i = 1, \ldots, n$$

yang ekivalen dengan persamaan:

$$AW = nW$$

Dalam teori matriks, formula tersebut menyatakan bahwa *W* adalah *eigen vector* dari matriks A dengan *eigenvalue n.* Bila ditulis secara lengkap maka persamaan tersebut akan terlihat pada Gambar IV.5.

$$\begin{bmatrix} \frac{w_1}{w_1} & \frac{w_1}{w_2} & \cdots & \frac{w_1}{w_n} \\ \frac{w_2}{w_1} & \frac{w_2}{w_2} & \cdots & \frac{w_2}{w_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{w_n}{w_1} & \frac{w_n}{w_2} & \cdots & \frac{w_n}{w_n} \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_n \end{bmatrix} = n. \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_n \end{bmatrix}$$

Gambar IV.5 Persamaan Matriks

Variabel n pada persamaan $AW = nW$ dapat digantikan secara umum dengan sebuah vektor λ, sehingga terbentuk persamaan sebagai berikut:

$$AW = \lambda W \qquad \ldots \text{(IV.6)}$$

dimana: $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)$

Setiap λ_n yang memenuhi persamaan (IV.6) dinamakan sebagai *eigenvalue*, sedangkan vektor W yang memenuhi persamaan (IV.6) tersebut dinamakan sebagai *eigenvector*.

Apabila dihubungkan dengan tahap 3, dan mengingat adanya teori matriks, maka:

1 Jika $\lambda_1, \lambda_2, \ldots, \lambda_n$ adalah *eigenvalue* dari A, dan karena matriks A adalah suatu matriks resiprokal dengan nilai $a_{ii} = 1$ untuk semua i, maka:

$$\sum_{i=1}^{n} \lambda_i = n = \text{jumlah elemen-elemen diagonal matriks } A,$$

artinya, apabila matriks A adalah matriks yang konsisten maka semua *eigenvalue* bernilai nol, kecuali satu yang bernilai sama dengan λ_{maks}. Bila matriks A adalah matriks yang tidak konsisten, variasi kecil atas a_{ij} akan membuat nilai *eigenvalue* terbesar, λ_{maks}, tetap dekat dengan n, dengan nilai *eigenvalue* lainnya mendekati nol.

2 Kesalahan kecil pada koefisien matriks a_{ij} akan menyebabkan penyimpangan yang kecil pula pada *eigenvalue*, oleh karena itu, untuk mendapatkan besarnya *eigenvector*, harus diselesaikan persamaan berikut:

$$AW = \lambda_{maks} . W$$

Nilai λ_{maks} dapat diperoleh dengan persamaan (IV.6) atau:

$$(A - \lambda_{maks} . I) W = O$$

dimana *I* adalah matriks identitas dan *O* adalah matriks nol

Nilai *eigenvector w* dapat diperoleh dengan mensubstitusikan nilai λ_{maks} ke dalam persamaan $(A - \lambda_{maks} . I) W = O$.

1.6 Sintesa Prioritas

Pada tahap sintesa prioritas dilakukan perhitungan bobot prioritas, dengan dua jenis prioritas, yaitu:

1 Prioritas Lokal

Prioritas lokal ditunjukkan sebagai himpunan *eigenvector* dalam setiap matriks perbandingan berpasangan. Nilai ini menggambarkan pengaruh relatif himpunan elemen dalam matriks tersebut terhadap elemen pada level tepat di atasnya.

2 Prioritas Global

Setiap himpunan elemen pada suatu matriks perbandingan berpasangan dapat dihitung nilai prioritas globalnya yang menyatakan pengaruh relatif masing-masing elemen terhadap pencapaian tujuan (*goal*) pada level paling atas (*top level*).

1.7 Pengujian Konsistensi

Hubungan preferensi yang dilakukan pada dua elemen tidak mempunyai masalah konsistensi relasi. Bila elemen a_1 adalah dua kali lebih penting dari elemen a_2, maka elemen a_2 adalah $\frac{1}{2}$ kali pentingnya dari elemen a_1. Tetapi konsistensi seperti itu tidak selalu berlaku apabila terdapat banyak elemen yang

harus dibandingkan. Karena keterbatasan kemampuan numerik manusia, maka prioritas yang diberikan untuk sekumpulan elemen tidaklah selalu konsisten secara logis. Hal ini berkaitan dengan penerapan AHP, yaitu bahwa penilaian dalam AHP dilakukan berdasarkan pengalaman dan pemahaman yang bersifat kualitatif dan kuantitatif. Sehingga secara numerik, terdapat kemungkinan suatu rangkaian penilaian untuk menyimpang dari konsistensi logis.

Pada prakteknya, nilai a_{ij} akan menyimpang dari rasio w_i/w_j dan dengan demikian, persamaan sebelumnya tidak akan terpenuhi. Pada matriks yang konsisten, secara praktis $\lambda_{maks}=n$, sedangkan pada matriks tidak konsisten setiap variasi dari a_{ij} akan membawa perubahan pada nilai λ_{maks}. Deviasi λ_{maks} dari n merupakan suatu parameter *Consistency Index* (*CI*) yang dinyatakan sebagai berikut:

$$CI = \frac{(\lambda_{maks} - n)}{(n-1)} \qquad \ldots \text{(IV.7)}$$

Nilai *CI* tidak akan berarti apabila tidak terdapat patokan untuk menyatakan apakah *CI* menunjukkan suatu matriks yang konsisten. Patokan ini selanjutnya dinamakan sebagai *Random Index* (*RI*) yang diperoleh berdasarkan serangkaian perbandingan random atas 500 sampel. Suatu matriks yang dihasilkan dari perbandingan yang dilakukan secara acak merupakan suatu matriks yang mutlak tidak konsisten.

Perbandingan antara *CI* dengan *RI* akan diperoleh patokan untuk menentukan tingkat konsistensi penilaian suatu matriks, yang disebut sebagai *Ratio Consistency* (*RC*), dengan persamaan:

$$RC = \frac{CI}{RI} \qquad \ldots \text{(IV.8)}$$

Dari 500 sampel matriks acak, sengan skala perbandingan 1-9 untuk beberapa orde matriks, diperoleh nilai rata-rata *RI* seperti disajikan pada Tabel IV.2.

Tabel IV.2 Nilai Indeks Random

Orde Matriks	1	2	3	4	5	6	7	8	9	10
Indeks Random	0.00	0.00	0.52	0.90	1.12	1.24	1.32	1.41	1.45	1.49

Hasil penilaian suatu matriks perbandingan dalam pengolahan AHP adalah konsisten apabila nilai rasio konsistensi (*CR*) tidak lebih dari 0.10. Apabila $CR \leq 0.10$, maka hasil penilaian dapat diterima atau dipertanggungjawabkan. Jika tidak, maka pengambil keputusan harus meninjau ulang masalah dan merevisi matriks perbandingan berpasangan.

Pengujian pada persamaan (IV.8) dilakukan untuk matriks perbandingan yang didapatkan dari partisipan. Pengujian harus pula dilakukan untuk hirarki. Prinsipnya adalah dengan mengalikan semua nilai *CI* dengan bobot suatu kriteria yang menjadi acuan pada suatu matriks perbandingan berpasangan, dan kemudian menjumlahkannya. Jumlah tersebut kemudian dibandingkan dengan nilai yang diperoleh dengan cara yang sama tetapi untuk suatu matriks random. Hasil akhirnya berupa suatu parameter yang disebut dengan *Consistency Ratio of Hierarchy* (CRH), dengan persamaan sebagai berikut:

$$CRH = \frac{CIH}{RIH} \qquad \ldots \text{(IV.9)}$$

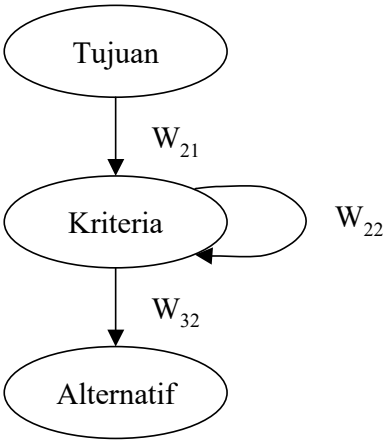
dimana: CIH = *Consistency Index of Hierarchy*
RIH = *Random Index of Hierarchy*

Secara rinci, prosedur perhitungan dapat diuraikan dengan langkah-langkah berikut:

1 Perbandingan antar elemen yang dilakukan untuk seluruh hirarki akan menghasilkan beberapa matriks berpasangan. Setiap matriks akan mempunyai beberapa hal berikut:
   a Satu kriteria yang menjadi acuan perbandingan antara kriteria pada tingkat hirarki di bawahnya.
   b Nilai bobot untuk kriteria acuan tersebut, relatif terhadap kriteria di tingkat lebih tinggi
   c Nilai *CI* untuk matriks perbandingan untuk matriks tersebut.
   d Nilai *RI* untuk matriks perbandingan untuk matriks tersebut.
2 Untuk setiap matriks perbandingan, kalikan nilai *CI* dengan bobot kriteria acuan. Jumlahkan semua hasil perkalian tersebut, maka didapatkan *Consistency Index of Hierarchy* (*CIH*).
3 Untuk setiap matriks perbandingan, kalikan nilai *RI* dengan bobot kriteria acuan. Jumlahkan semua hasil perkalian tersebut, maka didapatkan *Random Index of Hierarchy* (*RIH*).
4 Nilai CRH diperoleh dengan membagi *CIH* dengan *RIH*. Sama halnya dengan konsistensi matriks perbandingan berpasangan, suatu hirarki disebut konsisten apabila nilai *CRH* tidak lebih dari 0.10.

1.8 Dependensi antar Kriteria (Interdependensi)

Untuk memperoleh prioritas dalam sistem dengan pengaruh yang interdependen, penggunaan umpan balik merupakan ide yang umum digunakan pada hirarki. Dalam hal ini, elemen dari sistem digambarkan sebagai node dari jaringan.



Gambar IV.6 Hirarki dengan Interdependesi

Dua node dihubungkan dengan busur jika ada interaksi di antara keduanya. Untuk memperoleh prioritas global, vektor prioritas global dimasukkan dengan pendekatan matriks kolom dari pengaruh di antara elemen, yang dikenal sebagai supermatriks. Pembuatan matriks ini bertujuan untuk membatasi kekuasaan pengaruh hasil akumulatif dari tiap elemen dalam setiap elemen lain dengan interaksi. Penyajian matriks dalam tiga level digambarkan sebagai berikut:

$$W = \begin{array}{r} \\ \text{Tujuan } (G) \\ \text{Kriteria } (C) \\ \text{Alternatif } (A) \end{array} \begin{array}{c} \begin{array}{ccc} G & C & A \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ W_{21} & 0 & 0 \\ 0 & W_{32} & I \end{pmatrix} \end{array}$$

dimana: W_{21} dan W_{22} adalah matriks.

W_{21} sebenarnya merupakan vektor yang menggambarkan pengaruh dari tujuan pada kriteria, dan W_{32} menggambarkan pengaruh dari kriteria pada setiap alternatif. W merupakan supermatriks karena memasukkan matriks-matriks ke dalam matriks. Jika antar kriteria dependen, maka (2,2) dimasukkan dalam matriks W dituliskan sebagai W_{22} dan tidak bernilai nol, maka diperoleh:

$$W = \begin{pmatrix} 0 & 0 & 0 \\ W_{21} & W_{22} & 0 \\ 0 & W_{32} & I \end{pmatrix}$$

Sistem ini digambarkan dengan jaringan pada Gambar IV.6, dimana panah menggambarkan pengaruh dari elemen terhadap elemen lainnya dalam kelompok yang sama.

Pada supermatriks, semua perbandingan dibuat terhadap elemen yang memiliki pengaruh yang lebih. Secara bergantian, satu per satu dievaluasi elemen mana yang lebih berpengaruh. Dalam tipe masalah dependensi di antara dua kelompok, satu mengidentifikasi nilai kepentingan elemen dengan elemen lain, dan membandingkan secara berpasangan pengaruh dari setiap elemen. Salah satu keuntungan dari AHP adalah bahwa pertanyaan perbandingan berpasangan lebih cepat diinterpretasikan dengan interview.

W adalah matriks kolom stokastik, dan sintesa dari semua interaksi di antara elemen pada sistem ini dilambangkan W^{∞}. Jika matriks *irreducible* dan primitif, maka nilai limit yang dicapai W adalah tak terhingga, sebagai contoh: $W^{\infty} = \lim_{k \to \infty} W^{k}$. Nilai limit tersebut adalah unik, dan merupakan vektor kolom w^{∞}, dimana $W^{\infty} = w^{\infty} e^{T}$, dimana e^{T}=(1, …, 1). Jika W dihilangkan, kemudian dibutuhkan suatu nilai untuk dipertimbangkan n_1 dari prinsip eigenvalue adalah 1. Sebagai ilustrasi, ketika n_1=1, maka W^{∞} diperoleh:

$$W^{\infty} = (\text{I-W})^{-1}\ \Psi(1)/\ \Psi'(1)$$

Dimana $\Psi(\lambda)$ adalah polynomial minimum dari W dan $\Psi'(\lambda)$ adalah nilai derivatif awal yang berhubungan dengan λ. Sebagai contoh, untuk Gambar IV.6, n_1 diturunkan keluar untuk memiliki nilai n_1=1, maka:

$$W^{\infty} = \lim_{k \to \infty} \begin{pmatrix} 0 & 0 & 0 \\ W_{22}^{k} W_{21} & W_{22}^{k} & 0 \\ W_{32} \left(\sum_{h=0}^{k-2} W_{22}^{h} \right) & W_{32} \left(\sum_{h=0}^{k-1} W_{22}^{h} \right) & I \end{pmatrix}$$

Kini $|W_{22}|<1$ berpengaruh terhadap $(W_{22})^k$ cenderung menuju nilai nol, k cenderung menuju nilai tidak terbatas, sehingga:

$$W^{\infty} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ W_{32} (I - W_{22})^{-1} W_{21} & W_{32} (I - W_{22})^{-1} & I \end{pmatrix}$$

Kemudian, pengaruh dari tujuan pada urutan prioritas alternatif diberikan dengan nilai yang dimasukkan W^{∞}.

1.9 Pengambilan Keputusan Kelompok

Forman dan Peniwati (1998) menyatakan proses sintesis dengan metoda AHP tersebut dapat dilakukan dengan salah satu dari dua cara berikut:

- Metode aggregasi penilaian individu (*the aggregation of individual judgment/AIJ*), atau
- Metode aggregasi prioritas individu (*the aggregation of individual priorities/AIP*).

Untuk metode AIJ diasumsikan bahwa tiap anggota kelompok bertindak sebagai satu kesatuan dan tidak lagi bertindak berdasarkan identitas individu. Hasil penilaian tiap individu dalam kelompok diagregasikan dengan metode rata-rata geometrik. Untuk kasus ini, prinsip Pareto tidak relevan sehingga tidak perlu diperhatikan.

Dalam metode AIP, diasumsikan bahwa tiap anggota kelompok bertindak sebagai individu yang independen terhadap anggota kelompok lain. Urutan prioritas yang dihasilkan oleh tiap individu dalam kelompok diagregasikan dengan metode rata-rata geometrik atau rata-rata aritmetik, dan keduanya tidak melanggar prinsip Pareto.

Dalam menghasilkan preferensi kelompok dari preferensi individu, perlu diperhatikan terpenuhinya prinsip Pareto. Menurut prinsip Pareto, jika dua alternatif, alternatif a_1 dan alternatif a_2, dibandingkan dan tiap individu anggota kelompok lebih memilih alternatif a_1 dibandingkan dengan alternatif a_2, maka kelompok harus lebih memilih alternatif a_1 daripada alternatif a_2.

Zahir (1999) menyatakan bahwa dalam sebuah kelompok besar terdapat berbagai kemungkinan pola pemikiran, yaitu:

1 Seluruh anggota kelompok memiliki pemikiran yang sama
2 Pendapat anggota bervariasi tetapi mereka merupakan anggota suatu kelompok koheren yang homogen
3 Terdapat beberapa kluster atau sub-kelompok yang homogen dalam kelompok besar.

Kelompok yang homogen tidak menuntut preferensi yang identik dari masing-masing individu anggota kelompok. Dalam suatu kelompok yang homogen, pendapat masing-masing individu tetap bervariasi, namun memiliki "kesamaan". Kesamaan pendapat ini dapat diketahui dengan membandingkan nilai kosinus sudut yang dibentuk oleh pasangan preferensi individu anggota kelompok dengan batas kehomogenan (γ). Jika nilai kosinus sudut yang dibentuk oleh pasangan preferensi individu lebih besar atau sama dengan nilai batas kehomogenan (γ), maka pasangan individu ini dapat dikatakan memiliki "kesamaan".

1.9.1 *Techniques for* Analyzing *Consensus Relevant Data*/ACRD (Ngwenyama et. al., 1996)

Proses pengambilan keputusan dalam kelompok yang didukung komputer memungkinkan proses berlangsung anonim, dengan bantuan fasilitator sebagai pengarah diskusi. Untuk menghasilkan informasi bagi fasilitator, Ngwenyama et. al. (1996) mengajukan sejumlah teknik dan pendekatan untuk menganalisis data preferensi kelompok dalam proses pengambilan keputusan.

Teknik analisis data yang relevan dengan konsensus (*the techniques for analyzing consensus relevant data*/ACRD) yang diusulkan oleh Ngwenyama, et al. (1996) ini memanfaatkan data preferensi individual yang dihasilkan oleh tiap pengambil keputusan (nilai dan urutan) dalam kelompok terhadap sekumpulan alternatif keputusan dengan menggunakan metode AHP.

Analisis dilakukan berdasar pada kesamaan preferensi pasangan individu anggota kelompok, yang ditunjukkan oleh nilai kosinus sudut lebih besar atau sama dengan nilai batas kesepakatan (α). Ketidaksepakatan antar sepasang individu tercapai jika nilai kosinus sudut lebih kecil atau sama dengan nilai batas ketidaksepakatan (δ).

Nilai α yang mungkin adalah 0.985 (kosinus sudut 10^o) dan nilai δ yang mungkin adalah 0.966 (kosinus sudut 15^o). Rasionalisasi dari pengambilan sudut-sudut tersebut adalah bahwa sudut terbesar yang mungkin antara dua vektor bobot adalah 90^o. Sehingga 10^o pada skala 0^o-90^o ekivalen dengan 1 pada skala 1-9, dan 15^o ekivalen dengan 1.5 pada skala 1-9.

Secara konseptual, pendekatan untuk mendukung teknik ACRD ini dapat dibagi menjadi tiga tahapan proses, yaitu:

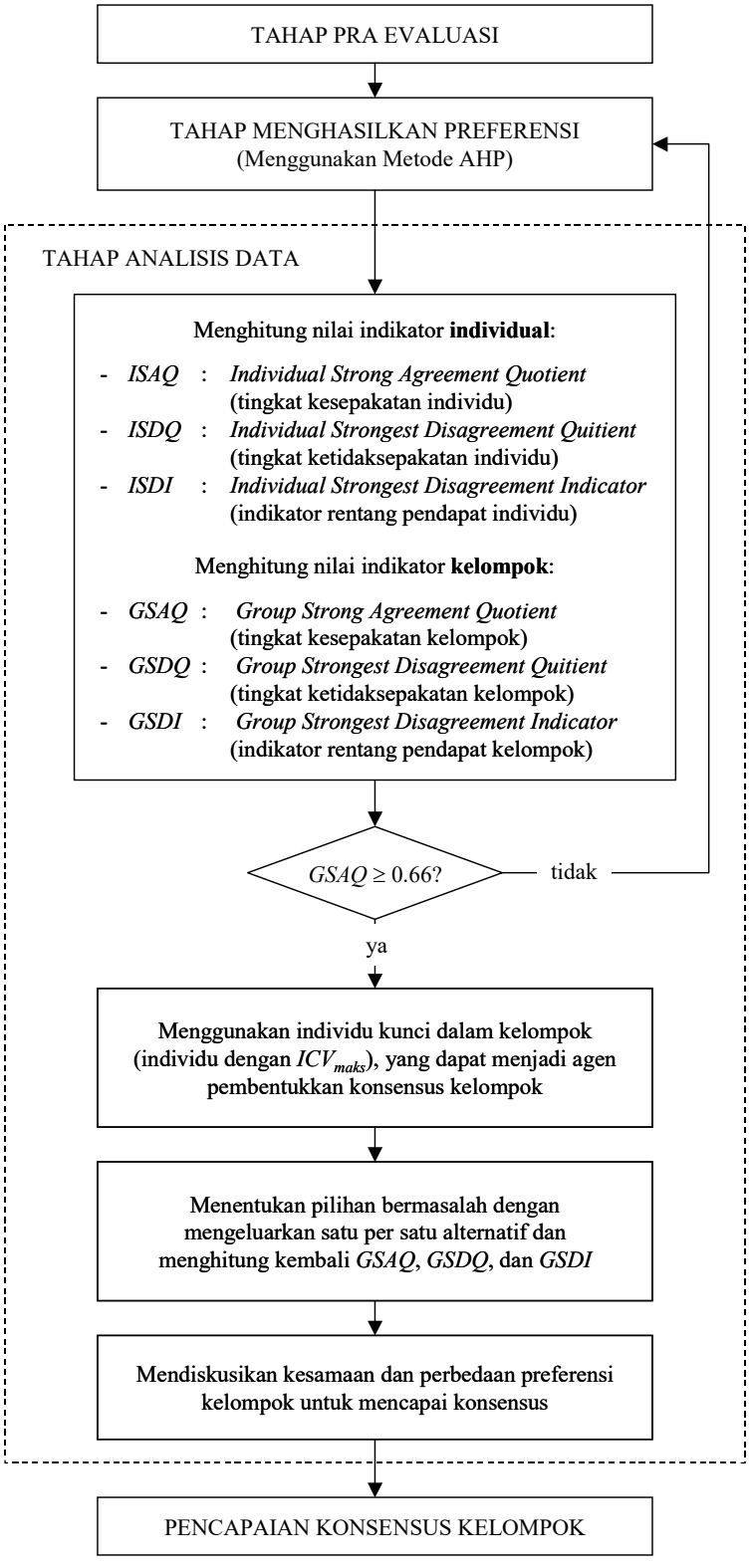
1 Tahap pra-evaluasi
2 Tahap menghasilkan preferensi
3 Tahap analisis data dan laporan

Tahap pra-evaluasi mencakup pemilihan alternatif untuk evaluasi dan penentuan kriteria evaluasi, serta penentuan batas kesepakatan untuk menentukan aturan penghentian proses pengambilan keputusan, berupa tercapainya suatu interval waktu yang dialokasikan atau tingkat kesepakatan tertentu terhadap isu (sebagian atau penuh). Sedangkan pada tahap menghasilkan preferensi dilakukan pengurutan alternatif dan penyajian data perbandingan dengan menggunakan metode AHP.

Pada tahap analisis data dan pelaporan dilakukan analisis terhadap data preferensi yang dilakukan oleh para pengambil keputusan untuk mengidentifikasi posisi mereka dalam proses pengambilan keputusan kelompok. Dalam tahap ini dilakukan juga identifikasi kemungkinan koalisi, identifikasi alternatif keputusan bermasalah, dan identifikasi individu kunci yang memiliki posisi preferensi yang memungkinkan negosiasi konsesnsus.

Pendekatan ini memungkinkan fasilitator dalam kelompok untuk menilai tingkat konsensus kelompok pada setiap tahap proses kelompok, sehingga informasi yang dihasilkan dapat membantu fasilitator menegosiasikan pembentukan konsensus dalam kelompok. Konsensus dapat dicapai saat peta konsensus telah diidentifikasi yang menggambarkan preferensi yang dapat diterima oleh kelompok. Dalam situasi ideal, sebaiknya dicapai kesepakatan total dalam kelompok. Namun umumnya, dengan mempertimbangkan perbedaan pendapat, hal ini tidak mungkin terjadi, sehingga dibutuhkan aturan penghentian.

Indikator individu yang dinyatakan oleh *Individual Consensus Vektor* (ICV^r), yang digunakan untuk mengidentifikasi individu yang memiliki tingkat kesepakatan yang baik dengan anggota kelompok lainnya dan tidak memiliki hambatan yang menyulitkan. Individu kunci ini memiliki nilai ICV^r terbesar (ICV^r_{maks}). Individu kunci digunakan untuk memfasilitasi pembentukkan konsensus kelompok.



Gambar IV.7 Proses Analisa Data ACRD

1.9.2 Algoritma Klaster Zahir (1999)

Untuk kelompok berukuran menengah (*intermidiate-sized group*) atau kelompok berukuran besar (*large group*), kehomogenan kelompok sulit dijamin atau dicapai, sehingga Zahir (1999) mengajukan suatu algoritma pengklasteran berdasarkan metode VAHP (*the Vector Space Formulation of the Analytic Hierarchy Process*). Dengan menggunakan algoritma ini, dalam suatu kelompok yang terdiri dari N anggota dapat dibentuk η klaster yang masing-masing homogen, dimana $1 \leq \eta \leq N$.

Klaster ditentukan secara alamiah dengan nilai batas keanggotaan klaster (γ). Besarnya nilai batas keanggotaan klaster (γ) bervariasi dan bergantung pada jenis permasalahannya. Nilai batas keanggotaan klaster ini ditetapkan berdasarkan kesepakatan anggota kelompok.

Untuk menentukan keanggotaan seorang pengambil keputusan terhadap suatu klaster, kosinus sudut antara vektor bobot pengambil keputusan tersebut dan vektor bobot resultan dari seluruh pengambil keputusan dalam klaster dibandingkan dengan nilai batas keanggotaan klaster (γ). Jika kosinus sudut antara vektor bobot pengambil keputusan tersebut dan vektor bobot resultan lebih besar atau sama dengan nilai batas keanggotaan klaster (γ), maka pengambil keputusan tersebut terpilih masuk menjadi anggota klaster. Demikian juga sebaliknya, jika kosinus sudut yang terbentuk antara vektor bobot pengambil keputusan tersebut dan vektor bobot resultan lebih kecil dari nilai batas keanggotaan klaster, maka pengambil keputusan tersebut tertunda untuk dipilih masuk menjadi anggota klaster.

Pengambil keputusan yang tertunda ini harus menunggu untuk kembali terpilih masuk klaster tersebut hingga terdapat pengambil keputusan lain dari kelompok yang terpilih masuk klaster. Jika tidak terpilih, pengambil keputusan yang tertunda tersebut harus menunggu untuk ditempatkan pada klaster lain.

Pembentukkan klaster pada algoritma Zahir (1999) ini menggunakan simulasi Monte Carlo.

2 Pengembangan Model

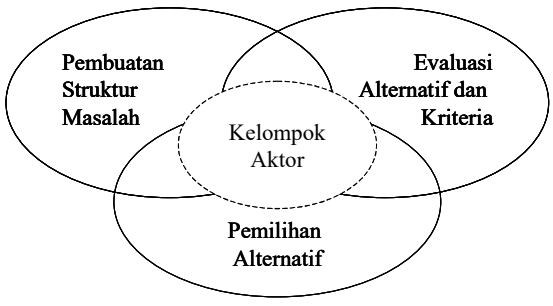
Pada bagian ini dibahas tentang tahapan pengembangan model, hal ini didasari oleh keinginan untuk menggambarkan sistematika dan metodologi pengembangan model.

Pengembangan model dalam penelitian ini adalah pengembangan model PKKM berbasis AHP. Hal ini juga didasari kenyataan bahwa AHP, dengan berbagai kelebihannya, merupakan model yang baik dalam menyelesaikan persoalan PKKM.

2.1 Perumusan Masalah

Perumusan masalah pada pengembangan model untuk penelitian ini dilakukan dengan melalu tiga tahap, yakni pengenalan masalah, identifikasi masalah, dan definisi masalah.

Sistematika metodologi pengembangan model PKKM harus mampu menciptakan tiga faktor penting dalam proses analisa pengambilan keputusan, yaitu (a) pembuatan struktur situasi permasalahan, (b) evaluasi alternatif dan kriteria, dan (c) pemilihan alternatif (Costa et. al., 1999)



Gambar IV.8 Faktor Penting dalam Analisa PKKM

Pengenalan Masalah

Pengenalan masalah dalam penelitian ini merupakan tahap observasi terhadap gejala-gejala (*symptoms*) dari objek penelitian. Penjajakan dilakukan dengan membangkitkan sebanyak mungkin pertanyaan-pertanyaan untuk memungkinkan munculnya gejala-gejala yang tersamar. Penafsiran pada gejala-gejala yang ada dilakukan berdasarkan sudut pandang peneliti terhadap permasalahan yang sedang diteliti.

Pengenalan masalah untuk pengembangan model keputusan didasari oleh pemikiran bahwa pada setiap saat, manusia selalu dihadapkan dengan masalah pengambilan keputusan, baik yang sederhana maupun yang rumit sekalipun. Sesederhana apapun suatu masalah pengambilan keputusan, manusia tetap melakukan suatu proses tertentu sampai didapatkannya suatu keputusan. Benar tidaknya atau baik tidaknya suatu keputusan akan sangat bergantung pada bagaimana seorang individu mampu mendayagunakan kemampuan berpikirnya dan sejauhmana individu tersebut memahami suatu permasalahan. Karena permasalahan di dunia nyata semakin kompleks dan makin sukar dibayangkan oleh kemampuan berpikir manusia, maka perlu dikembangkan metoda-metoda atau prosedur-prosedur pengambilan keputusan yang dapat mempermudah dan menambah keakuratan pengambilan keputusan.

Persoalan pengambilan keputusan merupakan bentuk optimasi dari pilihan-pilihan yang dibangun atas pertimbangan variabel-variabel kriteria dan/atau keterbatasan sumberdaya. Untuk memudahkan dan mempercepat proses pengambilan keputusan, dan didasarkan atas suatu metoda atau logika yang rasional, maka diperlukan suatu bentuk model yang mampu membantu pengambil keputusan memilih berbagai alternatif keputusan yang merupakan hasil pengolahan terhadap informasi-informasi yang diperoleh/tersedia. Pengembangan model yang dilakukan pada penelitian ini merupakan suatu upaya memperoleh model PKKM yang dapat membantu pengambil keputusan dalam menetapkan pilihan atas alternatif.

Identifikasi Masalah

Identifikasi masalah merupakan tahapan untuk mengenali masalah penyebab atau akar timbulnya gejala. Analisis dalam identifikasi masalah pada penelitian ini merupakan analitis situasional yang dilakukan untuk mengisolasi dan memahami variabel-variabel yang akan dilibatkan dalam model.

Kemampuan membuat keputusan tidak lepas dari kemampuan pengambil keputusan dalam mengelaborasi masalah, dan kemudian membuat analisa masalah. Berdasarkan pendekatan fisiologis manusia, pengambilan keputusan ini tidak lepas dari peran otak manusia, yang pada dasarnya terbagi atas dua bagian; otak kanan dan otak kiri. Otak kanan mencakup hal-hal yang berkaitan dengan perasaan manusia, pengalaman, naluri dan hal-hal lain yang berkaitan dengan emosi manusia. Secara singkat dapat dinyatakan bahwa otak kanan berkaitan dengan perasaan manusia. Sedangkan otak kiri manusia mencakup hal-hal yang sifatnya logis dan dapat dijelaskan dengan bukti yang kuat.

Dalam proses pengambilan keputusan, pada umumnya digunakan kombinasi dari otak kiri dan otak kanan, meskipun tidak dapat disangkal adanya kemungkinan bahwa banyak keputusan yang diambil hanya melalui satu kemampuan otak; baik otak kiri maupun otak kanan. Dengan kata lain, suatu proses pengambilan keputusan umumnya menggunakan kombinasi pendekatan rasional analitis dan intuitif emosional dalam penetapan keputusannya (Suryadi dan Ramdhani, 1998; Salusu, 1996; Jauch dan Glueck, 1995, Brodjonegoro, 1992).

Identifikasi masalah yang ditetapkan dalam penelitian ini, adalah kebutuhan kemudahan pengunaan PKKM. Model yang dikembangkan dipersyaratkan untuk memiliki kelebihan dalam hal fleksibilitas. Model ini diharapkan dapat memecahkan masalah yang kompleks dimana aspek atau kriteria yang diambil

cukup banyak, dengan struktur masalah yang belum jelas, serta ketidakpastian tersedianya data statistik yang akurat atau bahkan tidak ada sama sekali.

Sebab, adakalanya timbul masalah yang dirasakan dan diamati perlu diambil keputusan secepatnya, tetapi variasinya sedemikian rumit sehingga datanya tidak mungkin dapat dicatat secara numerik, tetapi hanya data kualitatif yang dapat diukur yaitu berdasarkan persepsi pengalaman atau intuisi. Dalam kondisi ini, masalah-masalah seperti perumusan perencanaan, konflik, proyeksi, alokasi sumberdaya, adalah beberapa dari banyak hal yang diharapkan dapat diselesaikan dengan baik oleh model yang dikembangkan.

Definisi Masalah

Definisi masalah pada pengembangan model dalam penelitian ini merupakan bentuk observasi. Hal ini dilakukan berdasarkan penelusuran model-model PKKM yang ada (studi pustaka) khususnya AHP dan topik-topik lainnya yang relevan dengan fokus penelitian.

Secara umum model yang dikembangkan harus mampu memberikan gambaran (*description*), memberikan penjelasan (*prescription*), dan memberikan perkiraan (*prediction*) dari realitas yang diselidiki. Dalam kaitan ini, karakteristik model yang dikembangkan memiliki ciri-ciri:

1 Tingkat fleksibilitas model yang tinggi
   Dapat menunjang pengambil keputusan di berbagai bidang fungsional (keuangan, pemasaran, operasi produksi, dan lain-lain);
2 Mekanisme transparansi
   Model memiliki kemampuan untuk menerangkan kembali (*rekonstruksi*) tanpa ada yang disembunyikan.
3 Interaksi model
   Memungkinkan para pembuat keputusan berinteraksi dengan model-model, termasuk memanipulasi model-model tersebut sesuai dengan kebutuhan

4 Fleksibilitas output

Mendukung para pembuat keputusan dengan menyediakan berbagai macam output, termasuk kemampuan grafik menyeluruh atas pertanyaan-pertanyaan pengandaian

Berdasarkan karakteristik model dikaitkan dengan persoalan PKKM, dengan berlandaskan identifikasi masalah, maka variabel-variabel yang dilibatkan dalam model terdiri atas:

1 Kriteria-kriteria
2 Alternatif-alternatif
3 Kendala (batasan) sumberdaya

Pengembangan model menggunakan prinsip-prinsip sebagai berikut:

1 Elaborasi

Pengembangan model dimulai dengan yang sederhana dan secara bertahap dielaborasi hingga memperoleh model yang lebih representatif. Penyederhanaan dilakukan dengan menggunakan sistem asumsi yang ketat, yang tercermin pada jumlah, sifat, dan relasi variabel-variabelnya.

2 Analogi

Pengembangan model dilakukan dengan menggunakan prinsip-prinsip hukum atau teori yang sudah dikenal secara meluas.

Berdasarkan prinsip-prinsip pengembangan model, metoda-metoda evaluasi yang dilibatkan dalam pengembangan model meliputi:

1 Model Hirarki Tujuan
2 Metoda pembobotan
3 Instrumen pengukuran interdependensi kriteria
4 Programa linier
5 Analisis sensitivitas
6 Analisis pembentukkan konsensus

Empat prinsip dijadikan sebagai titik pandang dalam penetapan informasi yang akan dilibatkan dalam model. Keempat prinsip tersebut adalah:

1 Keterorganisasian (*block building*)

Tujuan pengorganisasian proses pemodelan adalah untuk menyederhanakan spesifikasi interaksi di dalam sistem. Masing-masing blok menggambarkan satu bagian sistem yang bergantung pada beberapa atau sedikitnya satu variabel input, dan yang berubah menjadi variabel output. Maka sistem secara keseluruhan, dapat digambarkan dalam terminologi keterkaitan antar blok.

2 Relevansi (*relevance*)

Prinsip relevansi merupakan sifat yang melekat dalam model, karena model harus menggambarkan keadaan yang diamati. Dengan demikian, model hanya akan mencakup aspek-aspek yang relevan dengan sasaran-sasaran dan sudut pandang yang telah ditetapkan.

3 Keakuratan (*accuracy*)

Keakuratan analisis model merupakan pertimbangan utama dalam pengembangan model. Keakuratan tergantung pada tingkat kebutuhan penggunaan model terhadap persoalan yang diamati atau tergantung pada ketelitian yang diinginkan.

4 Tingkat Agregasi (*aggregation*)

Tingkat agregasi perlu dipertimbangkan sesuai dengan tingkat kecukupan atau kepuasan minimal yang harus didapat dengan memakai model. Maksudnya, sampai sejauhmana tiap-tiap komponen maupun aktifitas akan diteliti. Atau, komponen-komponen mana saja yang dapat dikelompokkan menjadi satu komponen yang lebih besar.

Berdasarkan prinsip-prinsip tersebut, data-data yang dibutuhkan pada saat pengambilan keputusan, terdiri atas:

1 Kriteria-kriteria

2 Alternatif-alternatif

3 Preferensi antar kriteria

4 Preferensi antar alternatif

5 Interdependensi kriteria

6 Kendala-kendala sumberdaya

2.2 Model Konseptual

Perumusan model konseptual dalam penelitian ini digunakan untuk menunjukkan keterkaitan antar variabel yang menentukan perilaku model. Perumusan model konseptual ini untuk menguraikan hubungan masalah, model, dan tujuan pengembangan.

Pendekatan yang dilakukan dalam pengembangan model adalah analisis permasalahan PKKM melalui prinsip: **dekomposisi**, **perbandingan penilaian**, **analisis interdependensi**, **sintesa prioritas**, **analisis pembentukkan konsesus,** dan **analisis pemilihan alternatif**.

Dekomposisi

Analisa masalah dimulai dengan merinci suatu keadaan yang kompleks atau tak berkerangka ke dalam komponen-komponennya, kemudian mengatur bagian-bagian dari komponen-komponen tersebut dalam bentuk hirarki.

Analisis Perbandingan

Analisis perbandingan pada prinsipnya dilakukan dengan memberikan bobot pada alternatif. Evaluasi pembobotan alternatif dilakukan dengan menggunakan perbandingan preferensi antar elemen keputusan. Analisis perbandingan dilakukan dengan memberikan bobot verbal atau numerik pada variabel yang dianggap penting dan pada akhirnya melakukan sintesa dari pendapat tadi untuk menentukan variabel mana yang memiliki prioritas tertinggi yang keluar sebagai hasil analisis.

Karena menggunakan input persepsi manusia, agar pemecahan masalahnya dapat menghasilkan sebuah keputusan yang baik, maka relevansi seorang responden dengan masalahnya sangat kuat atau memenuhi kriteria ahli (*expert*) di bidang masalah yang tengah diteliti.

Analisis Interdependensi

Pada dunia nyata dapat dilihat bahwa masing-masing kriteria ataupun sub kriteria dimungkinkan memiliki ketergantungan (interdependensi). Apabila antar kriteria memiliki hubungan interdependensi, maka perlu dilakukan penyesuaian nilai bobot terhadap masing-masing kriteria tersebut, sesuai dengan tingkat ketergantungannya.

Untuk itu maka, model menyediakan instrumen yang mampu mengukur interdependensi kriteria. Nilai interdependensi antar kriteria dinyatakan ruang dalam geometrik sebagai bentuk interseksi (irisan) antar kriteria.

Sintesa Prioritas

Sintesa prioritas dilakukan dengan membuat bobot pada masing-masing elemen hirarki tujuan. Pembobotan ini dilakukan atas dasar pertimbangan tingkat kepentingan kriteria yang berbeda untuk suatu persoalan, sintesa prioritas dilakukan dengan menghitung bobot (lokal dan global) kriteria dan alternatif.

Sebuah model yang baik dipersyaratkan untuk peka terhadap perubahan lingkungan, dan harus mampu mendukung pembuat keputusan dapat menjawab atas pertanyaan-pertanyaan pengandaian. Analisis sensitivitas merupakan suatu analisis yang disediakan oleh model.

Analisis Pembentukkan Konsensus

Model yang dikembangkan pada penelitian ini ditujukan untuk penggunaan individu maupun kelompok.

Untuk pengambilan keputusan kelompok, pengambilan keputusannya dilakukan oleh sekelompok individu yang dianggap layak untuk menentukan suatu keputusan. Keputusan kelompok dianggap lebih baik dibandingkan dengan pendapat perorangan.

Namun, pada sisi lain, AHP hanya membutuhkan satu jawaban untuk satu matriks perbandingan. Penilaian yang dilakukan oleh banyak partisipan akan memungkinkan untuk menghasilkan pendapat yang berbeda satu sama lainnya (Anonimus, 2000).

Preferensi kelompok dengan metode AHP dihasilkan dengan mensintesis preferensi tiap individu anggota kelompok. Namun, untuk melakukan sintesis preferensi individu ini terdapat prasyarat bahwa kelompok pengambil keputusan terjamin homogen (Zahir, 1999).

Analisis dalam pencapaian konsensus kelompok dapat dilakukan dengan memperhatikan "kesamaan" pendapat pasangan individu anggota kelompok partisan/responden, pendapat individu-individu tersebut dapat dimasukkan ke dalam klaster-klaster yang homogen.

Pembentukkan struktur klaster dapat dibuat dengan memanfaatkan informasi tentang kesepakatan individu, tingkat ketidaksepakatan individu, dan rentang ketidaksepakatan individu yang akan dimasukkan ke dalam suatu klaster. Struktur klaster yang dihasilkan dapat memberikan informasi untuk mengidentifikasi kemungkinan terdapatnya pendapat marjinal dari individu anggota kelompok, yang dapat digunakan sebagai referensi bagi anggota kelompok untuk memperoleh konsensus kelompok.

Struktur klaster dapat digunakan untuk mengidentifikasi objek keputusan bermasalah baik kriteria maupun alternatif, dengan memanfaatkan nilai agregat

kosinus pada objek tersebut. Berdasarkan informasi tersebut, untuk objek yang memiliki bobot rendah, dapat dilakukan kajian tentang kemungkinan penghilangan elemen tersebut dari proses analisis keputusan.

Analisis Pemilihan Alternatif

Pada beberapa kasus, suatu masalah tidak dapat menyelesaikan solusi berupa prioritas. Pada tahap implementasinya, masing-masing alternatif mungkin menghadapi persoalan keterbatasan (kendala-kendala) sumberdaya.

Untuk itu, pada bagian akhir analisa keputusan perlu dilakukan proses pengendalian output dengan menggunakan programa linier, dengan input berupa nilai dan variabel kendala (*constrain*) yang akan dihadapi dalam implementasi alternatif. Input untuk programa linier ini merupakan ukuran kriteria yang terukur dan operasional. Dalam perumusan masalahnya, programa linier menggunakan fungsi kendala yang secara faktual dapat menghambat proses pengambilan keputusan; misalnya keterbatasan sumberdaya, dan lain sebagainya. Penggunaan programa linier ini digunakan sebagai suatu instrumen untuk mengoperasionalkan suatu keputusan.

Berdasarkan perumusan masalah, dikaitkan dengan tujuan pemodelan, model konseptual pengembangan model AHP ini mengikuti tahapan sebagai berikut:

1 Mendefinisikan masalah dan menentukan bentuk solusi yang diinginkan;
2 Membuat struktur hirarki yang diawali dengan tujuan umum, dilanjutkan dengan subtujuan-subtujuan (kriteria) dan alternatif-alternatif pada tingkatan hirarki yang paling bawah;
3 Merumuskan kuisioner dan menetapkan responden;
4 Respoden membuat matriks perbandingan berpasangan yang menggambarkan kontribusi relatif atau pengaruh setiap elemen terhadap masing-masing tujuan atau kriteria yang setingkat di atasnya. Perbandingan dilakukan berdasarkan penilaian dari responden dengan

menilai tingkat kepentingan suatu elemen dibandingkan terhadap elemen lainnya;

5 Melakukan perbandingan berpasangan sehingga diperoleh penilaian seluruhnya sebanyak nx[(n-1)/2] buah, dengan n adalah banyaknya elemen yang dibandingkan.

6 Menghitung *eigenvalue* dan menguji konsistensinya, jika tidak konsisten maka pengambilan data diulangi.

7 Menghitung *eigenvector* dari setiap matriks perbandingan berpasangan. Nilai *eigenvector* merupakan bobot setiap elemen.

8 Memberikan penilaian interdependensi kriteria;

9 Mengulangi langkah 4, 5, 6, 7, dan 8 untuk seluruh tingkat hirarki.

10 Menghitung nilai bobot untuk masing-masing kriteria dengan atas pertimbangan interdependensi;

11 Menghitung sintesa prioritas dalam bentuk bobot lokal dan global;

12 Membuat rekapitulasi data yang diperoleh dari responden-responden dan melakukan analisis konsensus kelompok;

13 Menguji tingkat konsensus kelompok; apabila tidak terbentuk konsensus maka dilakukan proses pembentukkan konsensus, dan proses diawali lagi ke langkah 4 untuk masing-masing responden;

14 Mengevaluasi setiap alternatif berdasarkan kebutuhan, dan kendala-kendala operasional melalui model programa linier.

2.3 Formulasi Model

Formulasi model untuk prinsip **dekomposisi**, **perbandingan penilaian**, dan **sintesa prioritas** menggunakan formulasi model yang digunakan pada AHP. Fokus bahasan pada formulasi model pada penelitian ini adalah analisis interdependensi, analisis pembentukkan konsensus, dan analisis pemilihan alternatif.

2.3.1 Analisis Interdependensi Kriteria

Secara singkat, definisi interdependensi dapat dinyatakan apabila ditetapkan dua buah kriteria (c_1 dan c_2); c_1 dan c_2 adalah independen apabila c_1 tidak memberikan pengaruh terhadap c_2, dan begitu pula sebaliknya. Jika c_1 dan c_2 tidak bersifat independen, maka hubungan antara dua elemen ini adalah interdependen.

Menganalisis kemungkinan adanya interdependesi antar kriteria dapat diduga berdasarkan analisis korelasi antar kriteria. Analisis korelasi adalah analisis yang mempelajari derajat hubungan korelasi antara kriteria, dengan hasil analisis berupa koefisien korelasi. Dalam menghitung koefisien korelasi antara variabel c_i dan c_j dalam panduan ini digunakan koefisien korelasi Pearson (R), dengan persamaan matematis di bawah ini:

$$R_{C_i C_j} = \frac{n\sum c_i\, c_j - \left(\sum C_i\right)\left(\sum C_j\right)}{\sqrt{\{\, n.\ \sum c_i^{\,2} - (\sum c_i)^2\}\ .\ \{\, n.\ \sum c_j^{\,2} - (\sum c_j\ )^2\ \}}} \qquad \ldots(\text{IV.10})$$

dimana: $R_{C_i C_j}$ = korelasi antara kriteria-i dan kriteria-j

c_i = kriteria-i

c_j = kriteria-j

n = jumlah kriteria

Keeratan hubungan antar variabel tersebut dinyatakan dengan angka antara -1 dan 1, secara matematis dinyatakan sebagai $-1 < R_{C_i C_j} < 1$. Tanda hubungan yang negatif ($r < 0$) berarti terjadi konflik antar kriteria. Data yang digunakan untuk mengukur koefisien korelasi ini dapat menggunakan data internal (data preferensi).

Koefisien korelasi ini merupakan panduan pendugaan adanya interdependensi antar kriteria yang menyatakan dugaan bahwa semakin tinggi koefisien

korelasi antar kriteria maka semakin kuat dugaan adanya interdependensi antar kriteria.

Nilai interdependensi antar kriteria

Jika w_i adalah bobot kriteria ke-*i* yang merupakan bagian dari himpunan bobot kriteria (*W*), dimana kriteria tersebut merupakan bagian dari himpunan kriteria ($c_i \in C$) dan $w_i \geq 0$, serta jika:

(1) $(w_1 \cup w_2 \cup w_3 \cup ...) = w_1 + w_2 + w_3 + ...$ …(IV.11)

dimana $w_i \cap w_i = \emptyset$, $i \neq j$.

(2) $W = \sum_{i=1}^{n} w_i = 1$ …(IV.12)

dimana *n* adalah jumlah kriteria

Sehingga untuk setiap $c_i \in C$, maka berlaku hubungan:

$$w_i = 1 - w_i^* \quad \text{…(IV.13)}$$

Pembuktian persamaan (IV.13) ini adalah apabila W = $w_i \cup w_i^*$ dan $w_i \cap w_i^* = \emptyset$. Kemudian dengan meninjau persamaan (IV.11) dan (IV.12), maka 1= $w_i \cup w_i^*$. Persamaan (IV.13) terbukti.

Apabila w_1 dan w_2 adalah bagian dari *W*, maka berlaku hubungan:

$$w_1 \cup w_2 = w_1 + w_2 - (w_1 \cap w_2) \quad \text{…(IV.14)}$$

Pembuktian persamaan (IV.14) ini adalah bahwa untuk setiap $w_1 \cup w_2$, w_2 dapat dinyatakan dengan:

$w_1 \cup w_2 = w_1 \cup (w_1^* \cap w_2)$, dan

$w_2 = (w_1 \cap w_2) \cup (w_1^* \cap w_2)$

berdasarkan persamaan (IV.11), maka:

$$w_1 \cup w_2 = w_1 + (w_1^* \cap w_2) \quad \text{…(IV.15)}$$

$$w_2 = (w_1 \cap w_2) + (w_1^* \cap w_2) \quad \text{…(IV.16)}$$

bila persamaan (IV.16) merupakan jawaban untuk $(w_1^* \cap w_2)$ dan hasil ini disubstusikan pada persamaan (IV.15), maka diperoleh:

$$w_1 \cup w_2 = w_1 + w_2 - (w_1 \cap w_2)$$

Persamaan (IV.14) terbukti.

Sehingga untuk seluruh kriteria ($c_i \in C$) dengan i=1, 2, …, n, maka berlaku persamaan:

$$\bigcup_{n=1}^{n} w_i = \sum_{n=1}^{n} w_i - \bigcap_{n=1}^{n} w_i \quad \text{…(IV.17)}$$

Berdasarkan teorema-teorema di atas, maka interdependensi antar kriteria dapat dihitung dengan membuat perbandingan berpasangan antar dua kriteria yang digabungkan dengan suatu kriteria. Misalnya untuk tiga kriteria, berdasarkan metoda AHP, perbandingannya adalah sebagai berikut:

$$\begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \end{array} \begin{array}{c} \begin{array}{ccc} c_1 & c_2 & c_3 \end{array} \\ \begin{bmatrix} 1 & a_{12} & a_{13} \\ 1/a_{12} & 1 & a_{23} \\ 1/a_{13} & 1/a_{23} & 1 \end{bmatrix} \end{array}$$

yang menghasilkan bobot sebesar w_1, w_2, dan w_3, untuk memperoleh nilai interdependensi antara c_1 dan c_2 ($c_1 \cap c_2$), maka dilakukan perbandingan antara kriteria gabungan antara c_1 dan c_2 yang dinyatakan sebagai c'_{12} dengan c_3, sehingga diperoleh matriks perbandingan:

$$c'_{12} \begin{matrix} \\ c_3 \end{matrix} \begin{matrix} \begin{matrix} c'_{12} & c_3 \end{matrix} \\ \begin{bmatrix} 1 & a \\ 1/a & 1 \end{bmatrix} \end{matrix}$$

dengan hasil bobot sebesar w'_{12} dan w_3, berdasarkan persamaan (IV.14) diperoleh nilai interdependensi:

$$\begin{aligned} w_1 \cup w_2 &= w_1 + w_2 - (w_1 \cap w_2) \\ w_1 \cap w_2 &= w_1 + w_2 - (w_1 \cup w_2) \\ w_1 \cap w_2 &= w_1 + w_2 - w'_{12} \end{aligned} \qquad \ldots(\text{IV.18})$$

Untuk seluruh kriteria ($c_i \in C$) dengan i=1, 2, …, n, maka persamaan (IV.18) ekivalen dengan:

$$\bigcap_{i=1}^{n} w_i = \sum_{i=1}^{n} w_i - \bigcup_{i=1}^{n} w'_i \qquad \ldots(\text{IV.19})$$

Untuk memudahkan penggunaan model interdependensi ini, digunakan ruang geometrik. Interdependensi pada n kriteria untuk ruang geometrik ini dinyatakan untuk dalam ukuran ($w_i \cap w_i$). Bobot elemen baru dihitung dengan tetap mempertahankan rasio awal antara w_i dan w_j.

$$B_i = \frac{w_i}{w_i + w_j}\left[w_i + w_j - (w_i \cap w_j) \right] \quad \text{…(IV.20)}$$

Sehingga untuk seluruh kriteria ($c_i \in C$) dengan i=1, 2, …, n, maka berlaku persamaan:

$$B_i = \frac{w_i}{\sum_{j=1}^{n} w_j}\left[\sum_{i=1}^{n} w_j - \bigcap_{i=1}^{n} w_j \right] \quad \text{…(IV.21)}$$

dimana: B_i = bobot baru elemen ke-i

w_i = bobot awal elemen ke-i

w_i = bobot awal elemen ke-j yang memiliki karakteristik interdependen dengan elemen ke-i

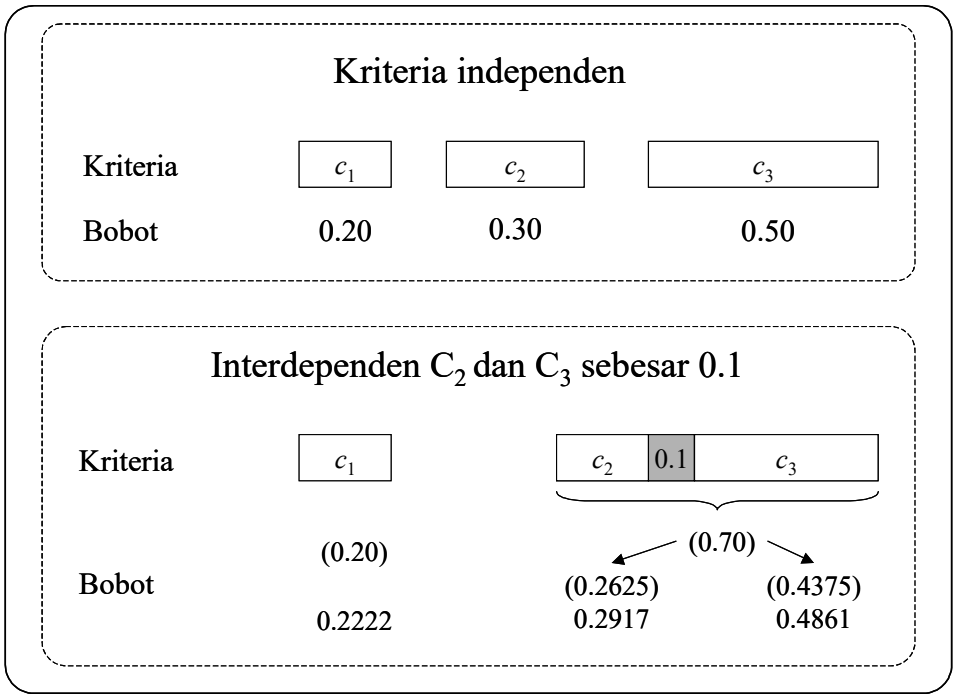
n = jumlah kriteria yang memiliki karakteristik interdependen dengan elemen ke-i

Normalisasi bobot dilakukan berdasarkan persamaan (IV.12). Sehingga bobot sintesa setiap elemen, dapat didefinisikan melalui persamaan:

$$\omega_i = \frac{B_i}{\sum_{j=1}^{n} B_j} \quad \text{…(IV.22)}$$

dimana : ω_i = Bobot akhir sintesa elemen ke-i

Sebagai ilustrasi untuk melihat permasalahan ini, dapat dilihat pada Gambar IV.9.



Gambar IV.9 Akibat Interdependensi Kriteria terhadap Perubahan Bobot

Untuk kasus ini, dapat diamati bahwa kriteria ketergantungan antara kriteria c_2 dan kriteria c_3 dapat mempengaruhi bobot c_1. Atau dengan kata lain, dengan adanya ketergantungan antara c_2 dan c_3 meningkatkan dominasi kriteria c_1. Untuk kasus yang sensitif dengan tingkat konflik kriteria yang tinggi, perubahan ini bisa jadi berpengaruh secara signifikan terhadap hasil keputusan.

Secara singkat, tahapan analisis interdependensi ini adalah:

1 Melakukan analisis untuk menduga adanya interdependensi antar kriteria;

2 Membuat perbandingan berpasangan yang dilakukan dengan membandingkan gabungan kriteria tersebut ($c_{ij}^{'}$) dengan kriteria lain.

3 Menghitung bobot $c_{ij}^{'}$ dan menetapkan nilai interdependensi

4 Menghitung bobot baru untuk masing-masing elemen

2.3.2 Sintesa Prioritas

Prioritas Lokal

Prioritas lokal ditunjukkan sebagai himpunan bobot elemen hasil perhitungan pada analisis pembobotan elemen dalam setiap matriks perbandingan berpasangan. Nilai ini menggambarkan pengaruh relatif himpunan elemen dalam matriks tersebut terhadap elemen pada level tepat di atasnya.

Prioritas Global

Setiap himpunan elemen pada suatu matriks perbandingan berpasangan dapat dihitung nilai prioritas globalnya yang menyatakan pengaruh relatif masing-masing elemen terhadap pencapaian tujuan (*goal*) pada level paling atas (*top level*). Penilaian bobot global mengikuti persamaan:

$$\varpi_i = \omega_i . \omega_j \qquad \ldots \text{(IV.23)}$$

dimana: ϖ_i = bobot global elemen ke-*i*

ω_i = bobot lokal elemen ke-*i*

ω_j = bobot gobal elemen ke-*j* yang tepat berada di atas elemen ke-*i*

Kecenderungan Alternatif

Nilai kecenderungan alternatif dinyatakan dalam bobot (global) yang merupakan penjumlahan simultan dari bobot global berbagai kriteria yang berada tepat satu hirarki di atas alternatif. Bobot masing-masing alternatif dihitung berdasarkan persamaan:

$$\mu_{a_i} = \sum_{j=1}^{n} \varpi_{a_i j} \qquad \ldots \text{(IV.24)}$$

dimana: μ_{a_i} = Bobot kecenderungan alternatif ke-*i*

$\varpi_{a_i j}$ = Bobot global alternatif ke-*i* berkenaan dengan elemen ke-*j* yang berada tepat di atasnya.

2.3.3 Analisis Sensitivitas

Untuk mempertahankan keandalan model dalam menghadapi berbagai perubahan, maka model menyediakan analisis sensitivitas dalam analisis keputusannya.

Analisis sensitivitas merupakan suatu analisa keputusan yang penting untuk dilibatkan dalam suatu model, karena melalui analisis sensitivitas pengambil keputusan dapat memperkirakan atau membuat antisipasi apabila ada sesuatu yang terjadi di luar perkiraan. Analisis sensitivitas digunakan untuk mengukur sejauhmana akibat perubahan bobot kriteria/subkriteria (variabel eksogen) terhadap bobot alternatif (variabel endogen).

Peningkatan nilai bobot pada salah satu variabel eksogen akan berimplikasi terhadap penurunan nilai variabel eksogen lainnya secara proporsional, hal yang sama berlaku untuk kondisi sebaliknya. Adapun perubahan nilai bobot ini dihitung berdasarkan persamaan:

$$\omega_i^P = \omega_i + \Delta\omega \qquad \text{…(IV.25)}$$

yang mengakibatkan perubahan variabel eksogen lainnya sebesar:

$$\omega_i^P = \omega_i - (\Delta\omega).\left(\frac{\omega_i}{\sum_{j=1}^{n}\omega_j}\right) \qquad \text{…(IV.26)}$$

Dimana: ω_i = bobot kriteria sebelum perubahan

ω_i^P = bobot kriteria setelah perubahan

$\Delta\omega$ = besarnya perubahan bobot kriteria

Perubahan yang terjadi untuk bobot alternatif (variabel endogen) mengikuti persamaan:

$$\mu_{a_i}^P = \sum_{j=1}^{n} \overline{\omega}_{a_i j}^P \qquad \text{…(IV.27)}$$

dimana: $\mu_{a_i}^P$ = Bobot kecenderungan alternatif ke-*i* setelah perubahan

$\overline{\omega}_{a_i j}^P$ = Bobot global alternatif ke-*i* berkenaan dengan perubahan bobot elemen ke-*j* yang berada tepat di atasnya.

Dalam SPKKM, analisis sensitivitas disajikan dalam dua bentuk, yaitu bentuk dialog yang menggunakan "trak" perubahan bobot dan implikasinya terhadap alternatif, dan analisis sensitivitas dalam bentuk grafik.

2.3.4 Analisis Konsensus pada Pengambilan Keputusan Kelompok

Pengembangan metode konsensus dilakukan untuk pengembangan keputusan kelompok berdasarkan konsep-konsep yang telah dikenal sebelumnya. Pengembangan metode konsensus ini dibagi pada empat tahapan utama, yaitu:

1 Tahap pra evaluasi

Pada tahap ini ditetapkan batasan kesepakatan yang membentuk klaster. Objek keputusan yang ditetapkan untuk setiap kriteria dan alternatif. Sedangkan batasan kesepakatan yang ditetapkan adalah nilai retang kesepakatan individu dalam kelompok (α), nilai ketidaksepakatan (δ), dan nilai batas keanggotaan klaster (γ), dan aturan penghentian, baik berupa batas waktu maupun tingkat konsensus yang dicapai.

2 Tahap menghasilkan preferensi

Pada tahap ini, preferensi individu tiap anggota kelompok terhadap objek keputusan dilakukan berdasarkan metoda AHP, termasuk pengujian konsistensinya.

3 Tahap analisa data

Pada tahap ini dilakukan penetapan nilai agregasi preferensi individu menjadi preferensi kelompok, analisis tingkat kesepakatan dan ketidaksepakatan individu, serta pembentukkan struktur klaster.

4 Tahap pembentukkan konsensus

Pembentukkan konsensus dilakukan metoda Delphi.

Preferensi kelompok.

Preferensi kelompok adalah nilai "rata-rata" preferensi individu. Preferensi kelompok dihitung berdasarkan metode AIP (Forman dan Peniwati, 1998), diasumsikan bahwa tiap anggota kelompok bertindak sebagai individu yang independen terhadap anggota kelompok lain. Urutan prioritas yang dihasilkan oleh tiap individu dalam kelompok diagregasikan dengan metode rata-rata rata-rata aritmetik.

$$\nu_i = \frac{\sum_{1=1}^{n} \omega_i}{n} \qquad \text{…(IV.28)}$$

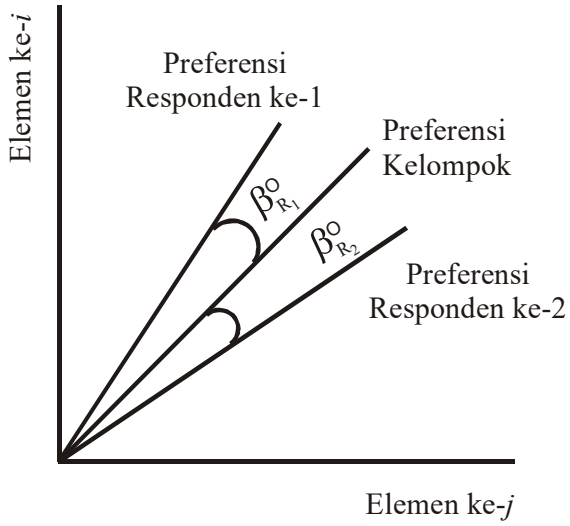
dimana: ν_i = bobot preferensi kelompok elemen ke-i

ω_i = bobot preferensi individu elemen ke-i

n = jumlah individu dalam kelompok

Untuk melihat hubungan antara nilai preferensi individu dengan preferensi kelompok ditentukan berdasarkan nilai besaran sudut $\beta_{R_i}^o$. Untuk memudahkan analisis maka dilakukan konversi nilai $\beta_{R_i}^o$ pada nilai cos $\beta_{R_i}^o$. Dengan ketentuan:

- cos $\beta_{R_i}^o$ mendekati nilai 1, menunjukkan kesepakatan yang kuat antara preferensi individu dengan preferensi kelompok
- cos $\beta_{R_i}^o$ mendekati nilai 0, menunjukkan kesepakatan yang lemah antara preferensi individu dengan preferensi kelompok



Gambar IV.10 Preferensi Individu dan Preferensi Kelompok dalam Ruang Vektor

Untuk keperluan analisa data, dilakukan penetapan tingkat kesepakatan dan ketidaksepakatan individu, serta pembentukkan struktur klaster.

Rentang kesepakatan (γ)

Rentang kesepakatan yang diambil pada penelitian ini adalah nilai rentang kesepakatan (γ) sebesar 1.0 pada skala AHP atau selisih 10^o pada ruang vektor. Dasar yang digunakan untuk penetapan rentang kesepakatan adalah rasionalisasi dari pengambilan sudut-sudut tersebut adalah bahwa sudut

terbesar yang mungkin antara dua vektor bobot adalah 90^o. Sehingga 10^o pada skala 0^o-90^o ekivalen dengan 1 pada skala 1-9, dan 15^o ekivalen dengan 1.5 pada skala 1-9 (Ngwenyama et. al., 1996).

Nilai batas kesepakatan (α) dan ketidaksepakatan (δ)

Nilai batas kesepakatan ditentukan sebesar α (untuk kesepakatan yang kuat) yang ditetapkan dengan nilai 0.985 (kosinus sudut 10^o) dan δ (untuk ketidaksepakatan yang kuat) ditetapkan dengan nilai 0.966 (kosinus sudut 15^o), maka anggota kelompok dikatakan memiliki kesepakatan yang kuat jika kosinus $\beta_{R_i}^o$ lebih kecil dari nilai α, dan ketidaksepakatan yang kuat jika kosinus $\beta_{R_i}^o$ lebih besar dari δ.

Pembentukkan struktur klaster

Struktur klaster menggunakan penetapan batas keanggotaan klaster (γ). Ketetapan pada pengembangan analisis konsensus pada penelitian ini menggunakan batas keanggotaan γ sebesar 10^o. Struktur klaster yang dirumuskan pada pengembangan model pada penelitian ini disajikan pada Tabel IV.3.

Tabel IV.3 Struktur Klaster

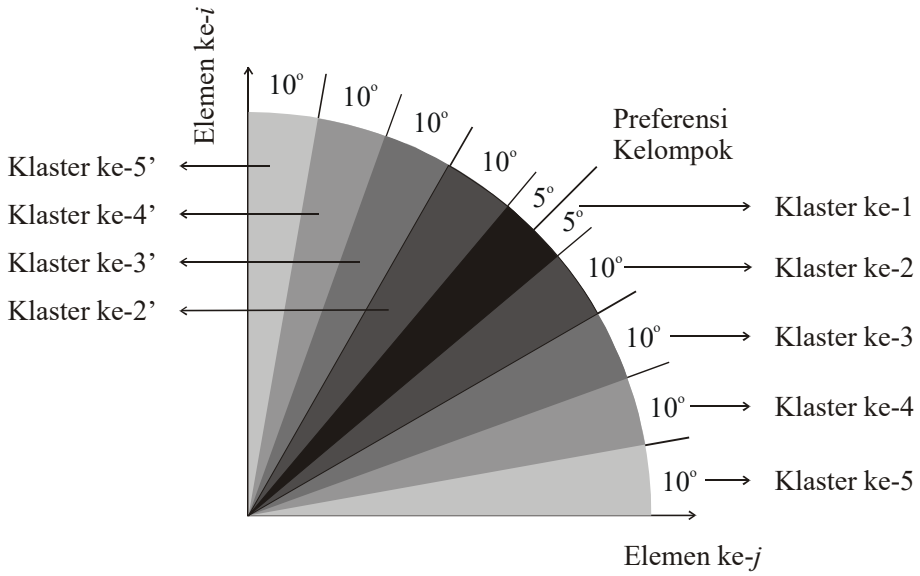
Klaster	Batasan $\beta_{R_i}^o$	Batasan kosinus $\beta_{R_i}^o$
Klaster ke-1	0º - 5 º	1.000 – 0.996
Klaster ke-2	5º - 15º	0.996 – 0.966
Klaster ke-2’		
Klaster ke-3	15º - 25º	0.966 – 0.906
Klaster ke-3’		
Klaster ke-4	25º - 35º	0.906 – 0.819
Klaster ke-4’		
Klaster ke-5	35º - 45º	0.819 – 0.707
Klaster ke-5’		
Klaster ke-6	45º - 55º	0.707 – 0.574
Klaster ke-6’		
Klaster ke-7	55º - 65º	0.574 – 0.423
Klaster ke-7’		
Klaster ke-8	65º - 75º	0.423 – 0.259
Klaster ke-8’		
Klaster ke-9	75º - 85º	0.259 – 0.087
Klaster ke-9’		
Klaster ke-10	85º - 90º	0.087 – 0.000
Klaster ke-10’		

Keterangan:

(a) Klaster ke-*n* menandakan ke arah sumbu mendatar

(b) Klaster ke-*n*’ menandakan ke arah sumbu tegak

Sebagai ilustrasi, seandainya preferensi kelompok menunjukkan bobot 0.5 untuk elemen ke-*i* dan 0.5 untuk elemen ke-*j*, maka struktur klaster untuk permasalahan ini ditunjukkan pada Gambar IV.11.



Gambar IV.11 Ruang Klaster Preferensi Individu dalam Kelompok

Koherensi preferensi individu ($\hat{P}_i$)

Untuk mengidentifikasi koherensi individu, analisis dilakukan dengan perhitungan koherensi preferensi individu ($\hat{P}_i$) berdasarkan nilai rata-rata cos $\beta_{R_i}^o$ untuk setiap elemen berdasarkan preferensi individu ke-*i*.

$$\hat{P}_i = \frac{\sum_{i=1}^{n} \cos \beta_i^o}{n} \qquad \ldots \text{(IV.29)}$$

dimana: $\hat{P}_i$: koherensi individu ke-*i*

$\beta_{R_i}^o$: sudut yang terbentuk antara preferensi individu dan kelompok

n : jumlah perbandingan preferensi yang berkaitan dengan elemen ke-*i*

Elemen yang memiliki nilai $\hat{P}_i \leq 0.966$ (δ) menunjukkan indikasi bahwa individu tersebut "bermasalah".

Koherensi elemen ($\hat{E}_i$)

Untuk mengidentifikasi objek keputusan bermasalah, analisis dilakukan dengan perhitungan koherensi elemen ($\hat{E}_i$) berdasarkan nilai rata-rata $\cos \beta_{R_i}^o$ untuk setiap preferensi individu yang berkaitan dengan elemen ke-*i*.

$$\hat{E}_i = \frac{\sum_{i=1}^{n} \left(\frac{\sum_{i=1}^{m} \cos \beta_i^o}{m} \right)}{n} \quad \ldots \text{(IV.30)}$$

dimana: $\hat{E}_i$: koherensi elemen ke-*i*

$\beta_{R_i}^o$: sudut yang terbentuk antara preferensi individu dan kelompok yang berkaitan dengan elemen ke-*i*

n : jumlah perbandingan preferensi yang berkaitan dengan elemen ke-*i*

m : jumlah individu anggota kelompok

Elemen yang memiliki nilai $\hat{E}_i \leq 0.966$ (δ) menunjukkan indikasi bahwa elemen (objek keputusan) tersebut "bermasalah". Objek keputusan bermasalah adalah objek keputusan yang jika dihilangkan dari vektor preferensi akan meningkatkan nilai aggregat koherensi elemen.

Aturan Penghentian

Aturan penghentian diskusi ditetapkan sebelum dilakukan penilaian terhadap objek keputusan. Aturan penghentian ini dapat dilakukan berupa suatu tingkat kesepakatan tertentu atau batasan waktu tertentu yang jika dicapai oleh

kelompok, maka kelompok harus melakukan sintesis terhadap keputusan individu walaupun tingkat kesepakatan belum dicapai.

Tingkat kesepakatan sebesar 0.66 atau lebih menunjukkan bahwa terdapat suara mayoritas dalam kelompok (lebih dari dua pertiga anggota kelompok memiliki tingkat kesepakatan yang sama). Hal ini berarti bahwa anggota kelompok memiliki urutan preferensi yang sangat dekat. Namun, jika tingkat kesepakatan (individu yang berada pada klaster 1) kurang dari 0.66, maka kelompok mengulangi lagi tahapan menghasilkan preferensi.

Pembentukkan Konsensus

Pembentukkan konsensus pada penelitian ini menggunakan kaidah-kaidah metoda Delphi. Diskusi (tukar-menukar informasi) dilakukan pertama kali dengan mendistribusikan pendapat yang marjinal pada preferensi kelompok. Iterasi kedua dilakukan dengan mendistribusikan pendapat yang memiliki nilai kedekatan yang paling kuat terhadap preferensi kelompok yang terbentuk pada tahap sebelumnya (setelah iterasi ke-1). Apabila konsensus tidak terbentuk setelah dua kali iterasi, fasilitator mengidentifikasi objek keputusan "bermasalah", kemudian melakukan diskusi dengan para anggota kelompok tentang kemungkinan dihilangkannya elemen tersebut. Iterasi berikutnya dilakukan dan berhenti sampai dijumpai preferensi yang homogen atau berdasarkan keterbatasan waktu.

Metoda konsensus yang dikembangkan pada penelitian ini dapat memberikan informasi mengenai tingkat kesepakatan dan ketidaksepakatan, serta rentang ketidaksepakatan individu dan kelompok, struktur klaster dan vektor preferensi klaster masing-masing, mengidentifikansi objek keputusan "bermasalah" dan pendapat marjinal.

2.3.5 Analisis Pemilihan Alternatif

Untuk kebutuhan-kebutuhan yang khusus, tidak semua permasalahan selesai dengan solusi dalam bentuk prioritas. Hal ini berkaitan dengan keterbatasan sumberdaya pada saat akan diimplementasikannya suatu alternatif. Untuk itu, pada bagian akhir model dilibatkan perhitungan *linear programing* (programa linier) untuk memilih alternatif yang paling layak dipilih.

Pemecahan masalah ini hanya sesuai untuk masalah-masalah yang memiliki asumsi-asumsi programa linier. Asumsi-asumsi yang harus dipenuhi oleh penggunaan programa linier pada analisis pemilihan alternatif ini adalah:

(1) Asumsi *proportionality* (kesebandingan)
Kontribusi setiap variabel keputusan terhadap fungsi tujuan adalah sebanding dengan nilai variabel keputusan.

(2) Asumsi *additivity* (penambahan)
Kontribusi setiap variabel keputusan terhadap fungsi tujuan tidak bergantung pada nilai dari variabel yang lain.

(3) Asumsi *certainty* (kepastian)
Setiap parameter diasumsikan dapat diketahui secara pasti.

Perumusan masalah untuk analisis pemilihan alternatif disajikan sebagai berikut:

Maksimasi $Z = \mu_{a_1}(a_1) + \mu_{a_2}(a_2) + \mu_{a_3}(a_3) + \ldots + \mu_{a_n}(a_n)$

S/t:
$$N_{11}\,a_1 + N_{12}\,a_2 + N_{13}\,a_3 + \ldots + N_{1n}\,a_n \leq K_1$$
$$N_{21}\,a_1 + N_{22}\,a_2 + N_{23}\,a_3 + \ldots + N_{2n}\,a_n \leq K_2$$

$$N_{m1}a_1 + N_{m2}\,a_2 + N_{m3}\,a_3 + \ldots + N_{mn}\,a_n \leq K_n \qquad \ldots(IV.31)$$

Dimana : Z = fungsi tujuan

μ_{a_i} = bobot kecenderungan alternatif ke-i

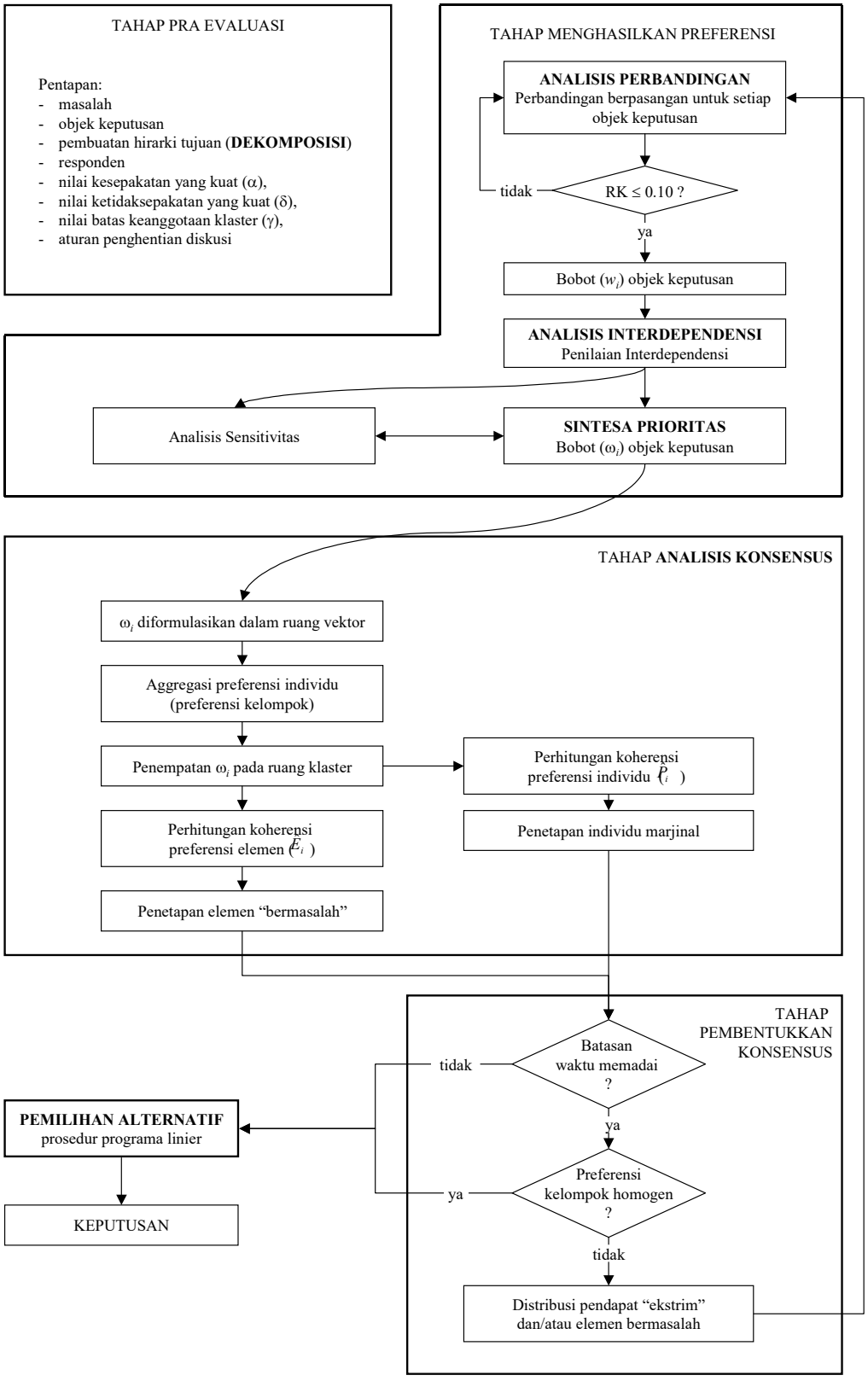
a_i = alternatif ke-i

K_i = nilai batasan ke-i (kendala/*constrain*)

N_{ij} = nilai implementasi ke-i untuk alternatif ke-j dengan pertimbangan K_i

Metoda programa linier yang digunakan pada model adalah metoda Simpleks untuk hasil keputusan dalam bilangan nyata, algoritma Branch and Bound untuk *integer programing*, dan algoritma enumerasi untuk keputusan dalam bentuk solusi 1-0.

Prosedur lengkap penyelesaian masalah untuk pengembangan model pada penelitian ini disajikan pada Gambar IV.12.



Gambar IV.12 Tahapan Penyelesaian Masalah

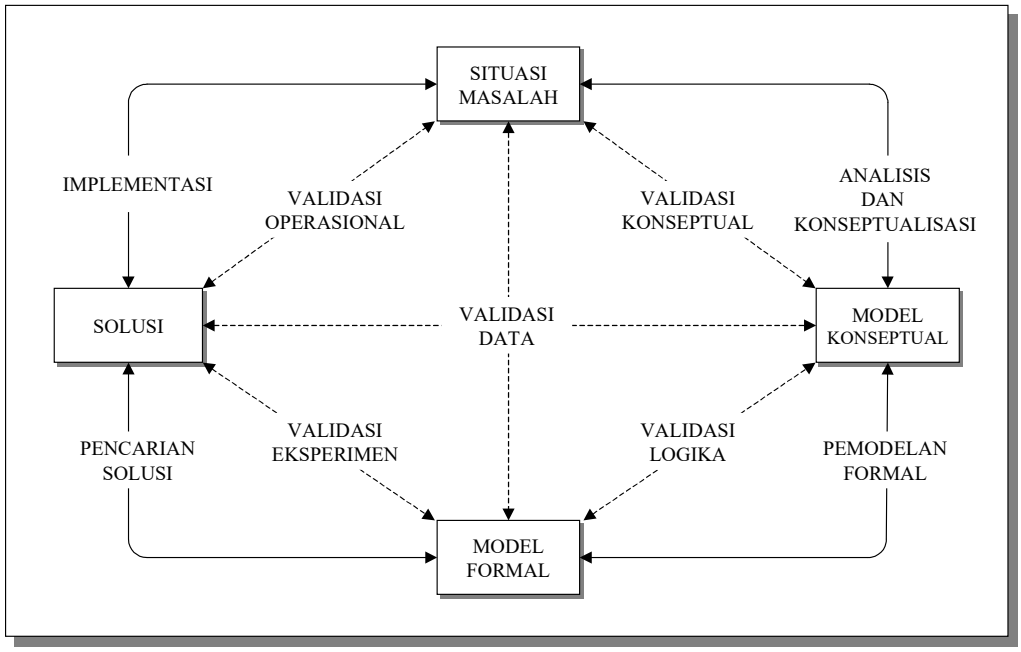
2.4 Validasi Model

Sebuah model dapat diterima sebagai sebuah model yang baik hanya apabila model tersebut berhasil melewati uji validasi. Aspek yang penting dalam pembuatan model dalam penelitian ini pada aplikasinya adalah pemilihan kriteria validasi yang sesuai yang mencapai kesesuaian antara tingkat kesesuaian model dan kompleksitas masalah dunia nyata.

Konsep validasi umumnya diukur berdasarkan tingkat kemanfaatan (*usefulness*), kemudahan penggunaan (*usability*), kemampuan representasi dari situasi permasalahan, dan pertimbangan biaya, serta variabel penting lainnya (Landry et. al., 1983). Kriteria yang digunakan untuk melihat kevalidan sistem diukur berdasarkan pertimbangan filosofi keilmuan, yang terdiri dari:

1 *Rasionalisme*, dengan tuntutan terhadap pertimbangan logis dari model.
2 *Empirisme*, dimana model dituntut untuk memiliki hubungan antara fakta.

Evaluasi validasi model yang digunakan dalam penelitian ini mengikuti alur proses validasi seperti tampak dalam Gambar IV.13.



Gambar IV.13 Validasi Model

Evaluasi atas validasi model menggunakan hubungan antara empat komponen utama, yang dinamakan situasi masalah, model konseptual, model formal dan solusi (rekomendasi) dari model.

Berbagai macam ketidakpuasan atas kinerja model, atau peluang penyempurnaan sebuah model menciptakan sebuah problem. Problematika ini kemudian diberikan sebual label "situasi masalah", yang pada prinsipnya mencoba mengikuti semua aspek yang terjadi pada dunia nyata (*real world*). Untuk itu, situasi masalah banyak dijumpai dan dibedakan atas kebutuhan, persepsi, dan perilaku dari aktor (pengembang model). Validasi sistem selalu dimulai dari pemahaman dan penggambaran atas situasi masalah ke dalam bentuk sebuah model.

2.4.1 Validasi Konseptual

Model konseptual merupakan penggambaran sebuah model melalui terminologi yang luas, suatu sudut pandang, tujuan yang dicapai, hubungan

antara variabel dari situasi masalah, tingkat aggregasi variabel elemen, dan hubungan antar variabel.

Derajat relevansi dari asumsi dan landasan teori merupakan acuan bagi model konseptual dari situasi masalah yang sesuai dengan keinginan pemakai dan penggunaan model merupakan inti dari validasi konseptual. Validasi konseptual mempersyaratkan pengembang model dan pembuat keputusan untuk meninjau kembali proses pembuatan model konseptual berdasarkan situasi masalah sampai mencapai suatu derajat validasi yang dapat diterima. Validasi konseptual diukur berdasarkan seberapa jauh model dapat memberikan penjelasan atas situasi masalah yang sedang dihadapi.

Sudut pandang dari model adalah mengukur bobot kepentingan alternatif dengan parameter penilaian preferensi pengambil keputusan berdasarkan kriteria-kriteria dan alternatif yang disusun secara berhirarki. Berdasarkan sudut pandang tersebut, diharapkan model mampu menjelaskan situasi masalah secara akurat. Model dibangun dengan menggunakan asumsi-asumsi bahwa masing-masing elemen satu level dalam hirarki bersifat homogen dan dapat dibandingkan, elemen-elemen yang menyusun model dianggap lengkap.

Tujuan yang dicapai dari rancangan awal model adalah sebuah keputusan implementasi sebuah rencana aktivitas organisasi dan mengukur seberapa jauh sebuah kriteria dapat mempengaruhi pemilihan alternatif tersebut. Model mampu menyelesaikan permasalahan yang diidentifikasikan pada rancangan awal model, hal ini menujukkan derajat validasi konseptual berdasarkan tujuan pengembangan sistem relatif memadai.

Hubungan antara variabel merupakan perilaku sistem yang menyederhanakan dari situasi masalah, hubungan ini digambarkan melalui penilaian alternatif berdasarkan kriteria. Konsep ini merupakan representasi model dari dunia

nyata bahwa untuk melihat masalah pemilihan alternatif umumnya menggunakan kriteria-kriteria untuk memudahkan evaluasi alternatif.

Evaluasi validitas konseptual model diduga relatif memadai, mengingat model hirarki kriteria dapat menjelaskan tentang faktor-faktor kunci yang digunakan sebagai alat ukur dalam pemilihan suatu alternatif. Hirarki model ini mampu menjelaskan tentang kerangka fungsional kriteria yang disusun sampai dengan tingkat yang operasional, dengan tujuan untuk memudahkan bagi pengguna sistem dalam melakukan penilaian atas kriteria-kriteria yang digunakan.

2.4.2 Validasi Logika

Validasi logika digunakan untuk menguji validitas sistem berdasarkan evaluasi atas model formal dalam melukiskan kebenaran dan akurasi dari situasi masalah yang ditetapkan dalam model konseptual. Dalam hal ini, model formal merupakan penterjemahan dari model konseptual dalam suatu simbol matematis. Tujuan pengembangan model formal ini adalah untuk memudahkan mempelajari masalah dan/atau untuk pencarian solusi.

Validasi logika merupakan pengujian atas hubungan antara variabel dalam model formal. Evaluasi validasi logika digunakan dalam pembentukan model formal dari model konseptual. Validasi logika ini digunakan dengan tujuan agar penterjemahan model konseptual ke dalam model formal berlangsung baik, sebab seringkali terjadi model formal kehilangan beberapa hal yang essensial dan substansial dari model konseptual karena kesulitan dalam proses penterjemahannya.

Evaluasi validitas logika model diduga relatif memadai, mengingat model ini mampu kembali situasi masalah secara benar, dan model memiliki mekanisme kerja model yang terukur. Beberapa persamaan atau teorema yang dikembangkan pada model dibuktikan secara analisis dan/atau pembuktian logika yang umum berlaku pada dunia nyata.

2.4.3 Validasi Eksperimental

Model formal umumnya dikembangkan untuk tujuan pencarian suatu solusi atas suatu masalah tertentu. Karakteristik dari model formal adalah teknik dan mekanisme pencarian solusi. Validasi eksperimental menunjukkan kualitas dan efisiensi dari mekanisme pemecahan masalah, yang meliputi evaluasi algoritma dalam eksperimen penggunaan sistem. Validasi eksperimental mempersyaratkan suatu sistem secara efisien dapat memecahkan masalah, dan perlu melibatkan analisis sensitivitas akibat dari perubahan beberapa parameter kunci.

Solusi merupakan salah satu ukuran validasi model. Solusi ini merupakan hasil dari model formal dengan suatu penggunakan teknik dan prosedur yang diperoleh dengan menggunakan beberapa contoh permasalahan di dunia nyata. Validasi eksperimental yang digunakan pada model menggunakan contoh hipotetik.

Tinjauan persyaratan validasi model berdasarkan validasi eksperimental pada model yang dilakukan dalam penelitian ini, menunjukkan bahwa model relatif dapat menyelesaikan permasalahan PKKM dengan baik.

2.4.4 Validasi Operasional

Kualitas dan derajat aplikatif dari sebuah solusi dan rekomendasinya ditentukan melalui validasi operasional, dengan perhatian pada pengguna dan kegunaan dari model formal dan hubungannya dengan situasi masalah. Validasi operasional digunakan untuk membantu pengambil keputusan dalam menerima atau menolak solusi dan rekomendasi model manakala model formal diimplementasikan. Hal ini mengindikasikan bahwa operasionalisasi suatu sistem harus tetap mempertimbangkan aspek waktu, usaha, dan biaya. Prosedur validasi operasional memberikan informasi hasil pengukuran atas keakuratan, kegunaan dan manfaat solusi dan rekomendasi model.

Mengingat model yang dikembangkan sebagai sebuah sistem yang menghasilkan suatu solusi yang terbatas (diantara himpunan alternatif yang telah ditetapkan, dan merupakan alternatif yang baik), maka dapat dipastikan bahwa model tidak akan mungkin menghasilkan suatu solusi alternatif yang buruk. Sehingga dapat ditarik kesimpulan bahwa model merupakan sistem yang memiliki derajat validasi operasional yang baik.

2.4.5 Validasi Data

Validasi data memainkan peranan penting dalam menjaga kecukupan, akurasi, ketepatan, dan kesesuaian data, dengan batas biaya yang diijinkan. Data merupakan input yang digunakan dalam pengembangan model konseptual dan formal, digunakan dalam analisis sensitivitas, dan bersyaratkan selalu terbaharukan.

Model yang membutuhkan data berupa kriteria, alternatif, preferensi antar elemen, dan keterbatasan sumberdaya. Keakuratan kriteria, alternatif, preferensi, interdependensi kriteria, dan keterbatasan sumberdaya tergantung dari kemampuan pengambil keputusan mengambil data. Namun, paling tidak, pada model disediakan suatu perangkat yang dapat mengukur tingkat keahlian dan/atau tingkat keseriusan responden dalam melakukan penilaian perbandingan, yaitu melalui analisis konsistensi.

BAB V
PERANCANGAN
SISTEM PENDUKUNG KEPUTUSAN KRITERIA MAJEMUK

Substansi kajian yang disajikan pada bab ini adalah sistematika perancangan SPKKM yang pada program aplikasinya diberi nama RATIONAL (*Ranking and Optimization in Choosing Alternatives*), sebagai instrumen untuk permasalahan PKKM, yang dibangun dari pengembangan model yang dituangkan pada Bab IV.

1 Analisis Sistem

Prinsip dasar yang menjadi acuan analisis sistem pada penelitian ini adalah bahwa setiap sistem diikat bersama oleh pertukaran informasi. Gagasan dasar analisa sistem perancangan RATIONAL adalah hubungan antara data, model, model dialog, jaringan komunikasi, serta keputusan yang dihasilkan. Titik awal pendekatannya adalah tujuan, dan fokusnya adalah pada rancangan SPKKM yang dibedakan dari rancangan komponen atau subsistem. Perancangan sistem untuk RATIONAL memanfaatkan analisis pengembangan dan pengelolaan sistem informasi dalam pengambilan keputusan.

Analisis sistem merupakan bagian penting dalam melakukan pengembangan sistem, sebelum dirumuskan ke dalam suatu kerangka desain sistem. Tujuan umum dari analisis sistem adalah menempatkan sistem, dengan segala keterbatasan yang ada, pada posisi yang paling baik, agar mampu melaksanakan fungsinya secara optimal. Analisa sistem dalam penelitian ini dimaksudkan untuk merumuskan hal-hal yang dipergunakan sebagai acuan dalam merancang sistem.

Analisis sistem merupakan penguraian dari suatu sistem yang utuh ke dalam bagian-bagian komponennya untuk mengidentifikasi dan mengevaluasi permasalahan yang mungkin terjadi dan kebutuhan-kebutuhan sistem. Analisis

sistem terdiri dari kegiatan-kegiatan yang diarahkan pada perencanaan dan pengendalian. Perencanaan mencakup penetapan sasaran-sasaran, rencana fasilitas, dan pengembangan program untuk mengatasi kegiatan-kegiatan yang berbeda dan menetapkan strategi yang berhubungan dengan lingkungan permasalahannya. Pengendalian mencakup pelaksanaan rencana, yang berkaitan dengan aliran informasi dan umpan-balik.

Kajian dalam analisis sistem diawali dengan menggambarkan keadaan SPK yang diinginkan. Gambaran tersebut dinyatakan sebagai unjuk kerja (kinerja) RATIONAL. Unjuk kerja sistem dalam analisis sistem ini, melibatkan empat kriteria utama, yaitu bahwa sistem harus logis, interaktif, adaptif, dan operasional.

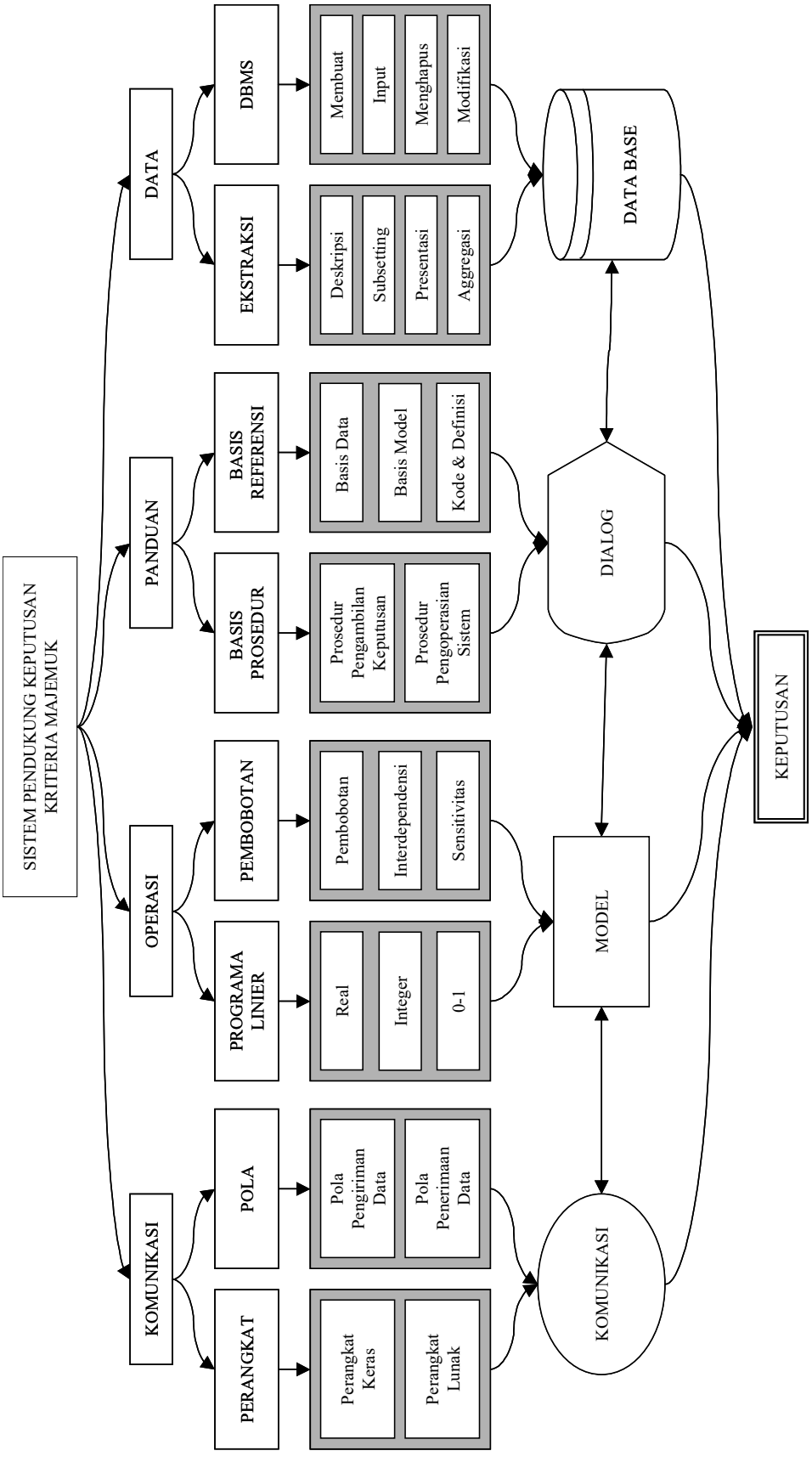
Pada dasarnya telaah strategis tidak dapat dianggap sebagai tujuan analisis sistem, karena yang patut dijadikan indikator pencapaian keberhasilan sistem, bukanlah hasil analisis, melainkan keluaran sistem yang sesuai dengan keinginan pengguna. Untuk itu, maka sistem perlu melibatkan suatu kriteria yang disebut sebagai hasil keputusan yang operasional. Ciri dari SPK adalah penggunaan yang mudah (*user freindly*), yang mungkin hanya bisa dibangun melalui modul-modul yang iteratif. SPK juga meniscayakan suatu persyaratan model yang dibangun adalah logis, yang sesuai (mirip) dengan berjalannya sistem pada dunia nyata. Salah satu sifat utama lainnya dari SPK yang dibangun adalah *adaptif* dan *fleksibel*, mengingat lingkungan, tugas-tugas dan pemakainya seringkali mengalami perubahan.

Salah satu sifat utama dari SPK yang dibangun adalah fleksibel, mengingat lingkungan, tugas-tugas dan pemakainya seringkali mengalami perubahan. Fleksibilitas RATIONAL dapat dikategorikan sebagai fleksibilitas F1 dimana sistem dapat memberikan kemampuan pada pemakai untuk menghadapi suatu masalah secara fleksibel dengan caranya sendiri. Fleksibilitas yang dimaksud ini adalah untuk menampilkan kemampuan pemahaman (*intelligence*),

perancangan (*design*), dan pemilihan (*choice*) serta kemampuan untuk menggali beberapa alternatif untuk menyelesaikan suatu masalah. Fleksibilitas yang tinggi pada RATIONAL diakibatkan oleh kebebasan pemakai untuk mengekspresikan kriteria dan alternatif dalam bentuk hirarki.

RATIONAL juga memiliki fleksibilitas tingkat F2 dengan kemampuan untuk memodifikasi konfigurasi sehingga dapat digunakan untuk menangani sekumpulan masalah yang diperluas. Fleksibilitas ini diimplementasikan dengan penambahan dan penghapusan representasi, operasi, bantuan memori dan mekanisme kontrol. Fleksibilitas ini merupakan kemampuan untuk menambah/menghapus sebuah grafik, atau peta, operasi pada sebuah grafik, atau pada menu pilihan.

Kerangka kerja perancangan komponen teknologi SPK didekati melalui sinergi antar elemen manajemen dialog, manajemen data, komponen model, dan jaringan komunikasi pada suatu sistem, pendekatan ini dilakukan berdasarkan kaidah efisiensi dan efektivitas setiap elemen sistem, sehingga setiap elemen dalam sistem indentik dan terkait secara penuh dalam sistem.



Gambar V.1 Rancangan Sistem Pendukung Keputusan Kriteria Majemuk

1.1 Manajemen Dialog

Komponen dialog dalam RATIONAL menggunakan perangkat keras dan perangkat lunak yang memberi sarana *interface* (antar muka) antara pemakai dengan sistem. Komponen dialog menyajikan output pada pemakai dan mengumpulkan input ke dalam sistem.

RATIONAL menggunakan komponen dialog yang mendukung perintah-perintah input dan output untuk berkomunikasi dengan pemakai. Perancangan komponen dialog ditujukan mendukung komunikasi yang efektif diantara pengambil keputusan dan sistem, perancangan dimulai dengan analisis proses pengambilan keputusan dan proses pengambilan keputusan. Analisis ini berfokus pada representasi (*output*) yang digunakan, dan mekanisme kontrol (*input*) yang digunakan untuk menjalankan operasi yang melibatkan representasi tersebut. Komponen dialog dirancang dengan menggunakan beberapa (kombinasi) model dialog.

1.2 Manajemen Basis Data

Manajemen basis data merupakan topik yang relevan dengan sebagian besar aplikasi komputer. Manajemen basis data dalam RATIONAL diaplikasikan untuk tujuan-tujuan:

1 Menyederhanakan pengumpulan dan pemeliharaan data yang digunakan;
2 Membatasi fungsi-fungsi pengelolaan yang dibutuhkan;
3 Menyederhanakan perancangan sistem;
4 Mengeliminasi performansi yang tidak perlu dan mendukung keamanan;
5 Meningkatkan kemampuan penggunaan data secara kolektif.

Sistem Manajemen Basis Data (DBMS~*Data Base Management System*) juga merupakan prasyarat penting bagi fleksibilitas RATIONAL, karena DBMS menangani pemeliharaan dan kontrol basis data, serta menyederhanakan program interface sistem dengan basis data.

Manajemen basis data merupakan komponen penting dari RATIONAL karena adanya perbedaan kebutuhan data. Basis data merupakan mekanisme integrasi berbagai jenis data internal dan eksternal. Ada kemungkinan data-data ini harus dimanipulasi (misalnya diaggregasi) atau dirubah (misalnya melalui struktur penyimpanan) dalam penggunaanya.

1.2.1 Model Data

Dalam pengembangan komponen manajemen basis data, akan terjadi pemilihan satu atau lebih model data. Model data adalah metoda penyajian, pengolahan, penyimpanan dan penanganan data dalam suatu komputer. Dalam RATIONAL, model data mempunyai tiga bagian, yaitu:

1 Kumpulan struktur data, yang terdiri dari sejumlah kriteria, himpunan alternatif, preferensi, nilai interdependesi kriteria, dan kendala teknis implementasi alternatif.

2 Kumpulan operasi yang dapat diterapkan pada struktur data, misalnya pembaharuan (*update*), pencarian informasi (*retrieval*), kombinasi dan sambungan (*summation*).

3 Kumpulan aturan/kendala yang menetapkan atau merubah status nilai pada struktur basis data, hal ini misalnya berlaku dalam penetapan batasan nilai preferensi.

Model data pada RATIONAL dibedakan dengan komponen pemodelan. Meskipun komponen manajemen basis data dapat digunakan oleh komponen pemodelan, model data merupakan model penyimpanan data dan operasi-operasi pada penyimpanan, sedangkan komponen pemodelan terdiri dari model-model keputusan yang bersifat permanen.

1.2.2 Ekstraksi Data

Ekstraksi data adalah suatu teknik untuk menghubungkan (*interfacing*) berbagai basis data sumber dengan basis data SPK. Teknik ini meliputi

penggabungan dan pemisahan dari basis data sumber untuk membentuk suatu data terekstraksi yang digunakan oleh komponen dialog dan pemodelan. Ekstraksi data pada RATIONAL terutama digunakan pada saat penggabungan nilai hasil kuisioner dari beberapa responden.

Empat macam operasi ekstraksi data yang termasuk fungsi sistem ekstraksi data yang digunakan pada RATIONAL yaitu:

1 Deskripsi data.

Operasi deskripsi data digunakan untuk menggambarkan file-file dalam basis data sumber

2 Pemilahan (*subsetting*).

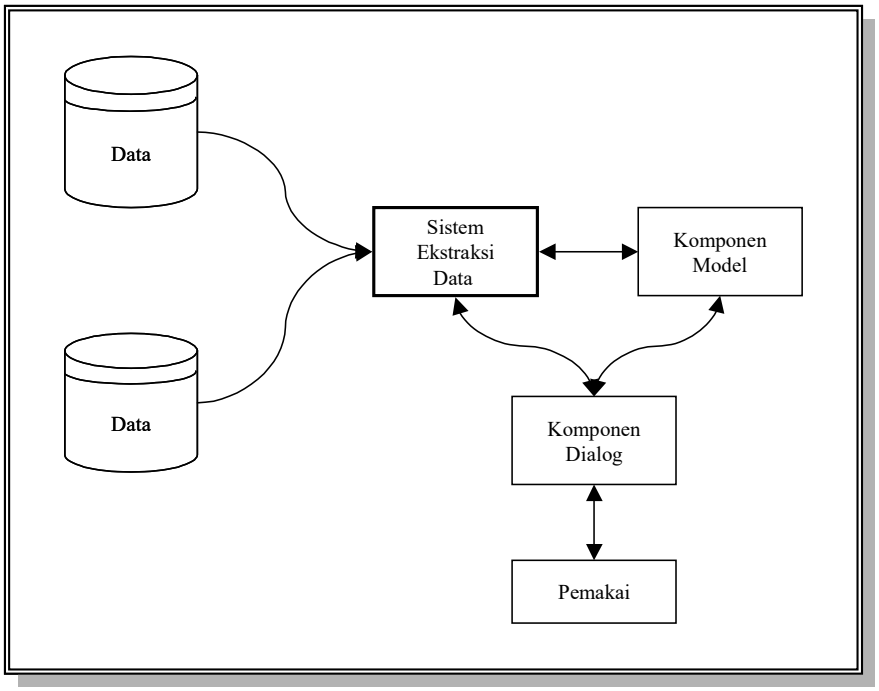
Operasi pemilahan dengan penggunaan kriteria aritmetik dan logik untuk memilih field atau record dari basis data sumber.

3 Agregasi.

Operasi agregasi memungkinkan menjumlah, menghitung, menggabungkan atau mengkombinasikan field atau record secara aritmetis.

4 Presentasi.

Operasi presentasi data dibutuhkan untuk melihat hasil-hasil operasi agregasi dan pemilahan.



Gambar V.2 Ikhtisar Ekstraksi Data

1.3 Manajemen Model

Komponen pemodelan memberikan kemampuan pengambil keputusan untuk menganalisis masalah melalui pengembangan dan perbandingan alternatif. Integrasi model-model ke dalam sistem berarti merubah sistem informasi manajemen (yang berdasarkan pendekatan komunikasi data dan pelaporan terintegrasi) menjadi suatu sistem pendukung keputusan.

Komponen pemodelan merupakan alat utama untuk mendukung aktivitas pengambilan keputusan dan pemecahan masalah. Komponen pemodelan mendukung aktivitas-aktivitas pada fase *design* (perancangan) dan *choice* (pemilihan), yang meliputi:

- Analisis, merupakan komponen model yang disajikan untuk melakukan operasi perhitungan pembobotan berdasarkan data kepentingan antar kriteria/alternatif dan ketergantungan antar kriteria;

- Proyeksi, dengan menyajikan analisis sensitivitas yang dibuat untuk menduga perubahan prioritas alternatif akibat dari perubahan bobot kriteria;
- Optimasi dengan melakukan beberapa operasi programa linier
- Penetapan alternatif, merupakan hasil akumulasi proses analisis dan optimasi.

Perancangan komponen pemodelan memungkinkan pengambil keputusan untuk mendukung aktivitas-aktivitas secara langsung. Beberapa kapabilitas yang dibutuhkan pemodelan dalam RATIONAL adalah sebagai berikut:

1 Interface
   a Pemakai dapat bekerja dalam proses pemecahan masalah tanpa selingan (*distraksi*) yang tidak perlu
   b Parameter kontrol diekspresikan dalam bentuk yang mudah dikenali oleh pemakai

2 Kontrol
   a Pemakai diberi suatu spektrum kontrol. Sistem mendukung operasi manual yang memungkinkan pemakai dapat memilih level operasi algoritma yang sesuai
   b Mekanisme kontrol memungkinkan pemahaman pemakai secara langsung terhadap solusi masalah.

3 Fleksibilitas
   Operasi-operasi manual dan algoritmik dapat saling dipertukarkan sehingga pemakai dapat mengembangkan sebagian solusi melalui metoda manual dan kemudian melanjutkannya dengan metoda algoritma, atau sebaliknya.

4 Umpan balik (*feed back*)
   a Sistem menyediakan umpan balik sehingga pemakai mengetahui secara penuh kedudukan proses solusi setiap saat.
   b Perancangan menggunakan sistem umpan balik

1.3.1 Basis Model

Basis model pada RATIONAL, terdiri dari model-model yang dibangun secara permanen dalam sistem, dan model-model khusus yang dibuat temporal (dalam hal ini bentuk hirarki tujuan). Dalam cara yang sama, basis model disimpan, ditangani dan dioperasikan dibawah kontrol *Model Base Management System* (MBMS) yang analog dengan DBMS. Empat fungsi umum yang paling penting adalah:

1 Pengembangan (*generation*), yaitu mekanisme fleksibel untuk pengembangan dan penurunan model-model;

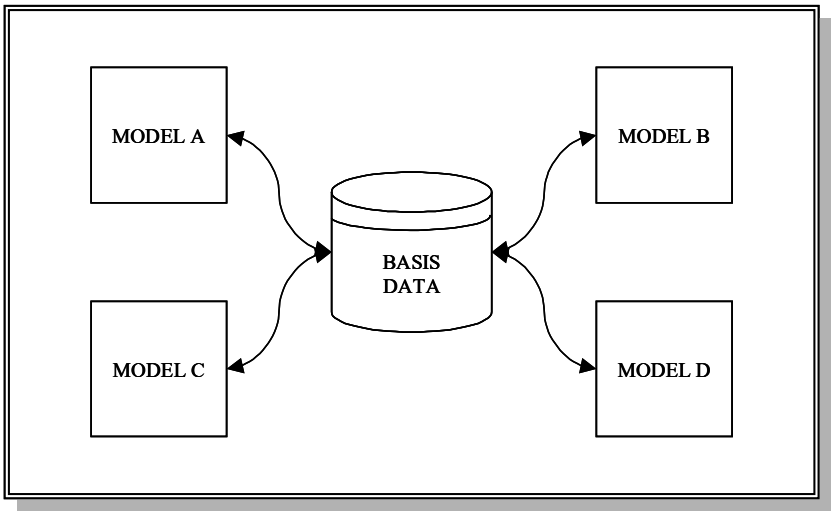
2 *Restrukturisasi*, yaitu suatu cara untuk menetapkan kembali atau restrukturisasi suatu model sebagai tanggapan terhadap perubahan dalam situasi yang dimodelkan;

3 *Update*, yaitu suatu prosedur untuk memperbaharui suatu model sebagai tanggapan terhadap perubahan data;

4 *Report generation-equiry*, yaitu operasi model untuk memperoleh pendukung keputusan yang diinginkan.

1.3.2 Integrasi dengan Komponen Dialog

Basis model dengan sistem manajemennya diintegrasikan dengan komponen dialog secara langsung, untuk memberikan kontrol langsung kepada pemakai selama operasi, manipulasi, dan penggunaan model-model.



Gambar V.3 Integrasi Model dengan Basis Data

Gambar V.3 menunjukkan beberapa model yang diintegrasikan dengan satu basis data. Setiap model memperoleh input dan nilai-nilai parameter dari basis data dan menghasilkan output kembali ke basis data tersebut. Dengan hubungan langsung demikian, model dapat diperbaiki, dimodifikasi atau direstrukturisasi.

1.4 Jaringan Komunikasi

SPKKM yang dirancang pada penelitian ini, menggunakan komponen jaringan komunikasi yang mendukung perintah-perintah penerimaan dan pengiriman data untuk berkomunikasi antar individu dalam kelompok pengambil keputusan. Perancangan komponen jaringan komunikasi ditujukan mendukung komunikasi yang efektif diantara mereka

Komponen jaringan komunikasi dalam RATIONAL tidak terintegrasi secara utuh dalam suatu perangkat lunak program, tetapi RATIONAL yang dirancang mendukung penyediaan sarana pengiriman dan penerimaan data dalam bentuk basis data yang terstruktur. Secara teknis, perangkat lunak maupun perangkat keras memanfaatkan teknologi yang sudah ada, yaitu jaringan internet dengan segala perangkatnya, khususnya untuk aplikasi *electronic mail*.

2 Rancangan Konstruksi Sistem

Perancangan sistem (*system design*) terbentuk setelah syarat-syarat analisis sistem dipenuhi. Perancangan sistem pada penelitian ini terdiri dari dua aktivitas utama, yaitu:

1 Peninjauan kebutuhan informasi dan fungsi sistem

2 Mengembangkan model untuk sistem, termasuk spesifikasi logis dan fisik, yang terdiri dari *output* (keluaran), *input* (masukan), *processing* (proses), *storage* (penyimpanan), dan *procedure* (prosedur).

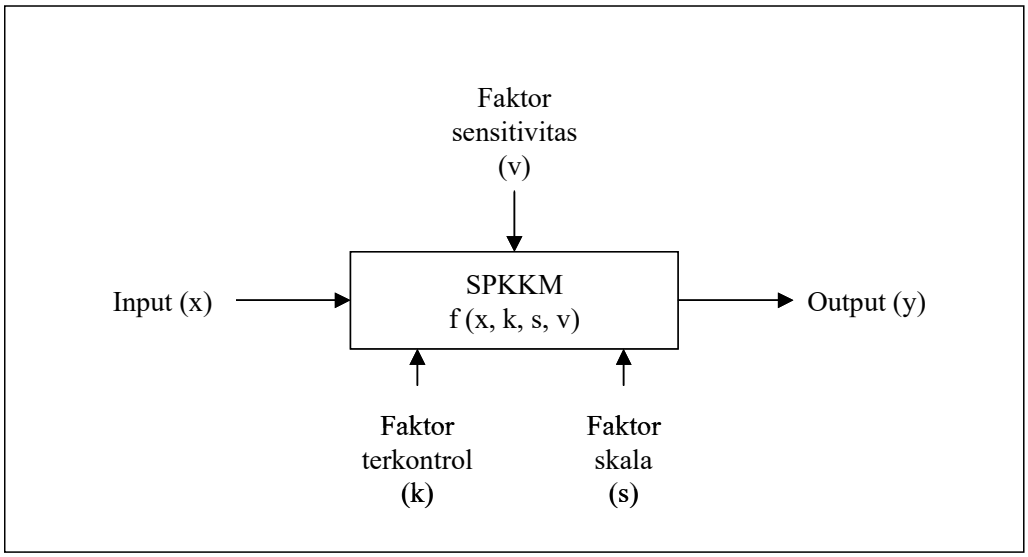
Selama fase ini, perancangan harus memenuhi tujuan, kemampuan dan konsep-konsep umum sistem. Oleh karena itu, setiap kebutuhan sistem harus ditinjau ulang sebelum memulai perancangan karena implementasi sistem akan menghadapi berbagai batasan-batasan. Untuk RATIONAL, rekomendasi yang diijinkan diakhir fase adalah sistem harus logis dan aplikabel. Artinya sistem harus menggunakan instrumen metodologik yang rasional dan mudah untuk dipelajari/digunakan dalam menghadapi persoalan yang akan dipecahkan.

Batasan-batasan tersebut diperhatikan selama perancangan sistem. Ini berarti tidak saja peninjauan sistem diperhatikan pada akhir fase perancangan, tetapi secara teratur diperhatikan agar sesuai batasan-batasan lain yang ditemukan setiap perkembangan perancangan.

Rancangan sistem disusun berdasarkan klasifikasi faktor yang diidentifikasikan berdasarkan elemen sistem yang dapat mempengaruhi kualitas (variabel respon) dari SPKKM yang dirancang. Klasifikasi faktor tersebut disajikan pada Gambar V.4.

- Faktor input merupakan sebuah representasi dari dunia nyata yang dinyatakan dengan parameter-parameter yang berkesuaian dengan klasifikasi data yang dapat diolah oleh RATIONAL;

- Faktor terkontrol adalah mekanisme pengendalian sistem melalui bentuk model yang terintegrasi dengan sistem yang parameter-parameter dikontrol pada saat rekayasa dan analisa sistem;
- Faktor skala adalah faktor yang berupa skala yang dibuat untuk memudahkan pemasukan input, dan interpretasi infomasi hasil pengolahan RATIONAL;
- Faktor sensitivitas adalah faktor yang dilibatkan dalam model sebagai parameter yang diduga akan menyebabkan terjadinya variasi (deviasi) karakteristik dari data yang menjadi input bagi RATIONAL;
- Respon merupakan informasi dan rekomendasi keputusan yang dihasilkan oleh RATIONAL berdasarkan fungsi dari faktor input, faktor terkontrol, faktor skala, dan faktor sensitivitas.



Gambar V.4 Klasifikasi Faktor dalam RATIONAL

Untuk menghindari kesalahan dari hasil perancangan, dan memudahkan di dalam proses pembuatannya terlebih dahulu dibuat prosedur pembuatan program secara ringkas dan terstruktur. Adapun prosedur tersebut adalah:

1 Analisis sistem yang ada (*exist*) dalam lingkup SPKKM, yang berkaitan dengan data yang diperlukan, cara pengumpulan data dan cara

pengolahan data sehingga menjadi informasi yang berguna bagi proses pengambilan keputusan;

2 Identifikasi jenis keputusan yang akan diambil oleh pemakai melalui sistem yang dirancang, dan menentukan bentuk representasi yang diinginkan, yang diharapkan mampu mendukung proses pengambilan keputusan;

3 Menentukan operasi-operasi yang diperlukan untuk dapat mewujudkan representasi yang diinginkan pemakai, sehingga memudahkan untuk menganalisa kebutuhan model yang dapat digunakan dalam proses pengambilan keputusan;

4 Menentukan perangkat lunak yang dapat digunakan dalam perancangan RATIONAL, yang diharapkan dapat memudahkan dalam pembuatan programnya, baik dalam proses pemasukan data, penyimpanan data, dan akses data, serta mungkin dalam menampilkan representasi dan operasi yang terlibat didalamnya;

5 Pembuatan program yang sesuai dengan mekanisme kontrol dan model dialog yang diinginkan pemakai;

6 Uji coba program tahap awal;

7 Evaluasi dan modifikasi program yang belum memenuhi kebutuhan yang diinginkan oleh pemakai;

8 Kompilasi program yang dibuat menjadi berekstensi .exe, sehingga dapat memudahkan dan mempercepat eksekusi program oleh pemakai;

9 Mengadaptasikan sistem yang telah dirancang sesuai dengan kebutuhan yang diinginkan, dan bila perlu dilakukan modifikasi model dan data base yang telah dibangun dalam sistem.

Perancangan spesifikasi RATIONAL dilakukan berdasarkan pendekatan *top-down* dalam implementasinya, dilakukan berdasarkan modul-modul pada tingkat tertinggi. Modul pada tingkat lebih tinggi ini akan diturunkan menjadi modul tingkat yang lebih rendah yang mampu memenuhi kebutuhan ditingkat bawah (Kowal, 1988).

Pendekatan ini berusaha mengembangkan model aliran informasi di dalam program dan merancang sistem informasi yang sesuai dengan aliran informasi tersebut. Metoda ini menggunakan pendekatan logis dalam pengembangan suatu rencana secara menyeluruh.

Pendekatan ini dimulai dari level atas program, yaitu level penentuan kebutuhan sistem. Pendekatan ini dimulai dengan mendefinisikan model konseptual performasi sistem yang dihasilkan. Langkah selanjutnya adalah dilakukannya analisis kebutuhan informasi. Setelah kebutuhan informasi ditentukan, dilanjutkan ke pemrosesan transaksi yaitu penentuan output, input, basis data, prosedur-prosedur operasi dan kontrol. Pendekatan ini juga merupakan ciri-ciri dari pendekatan terstruktur.

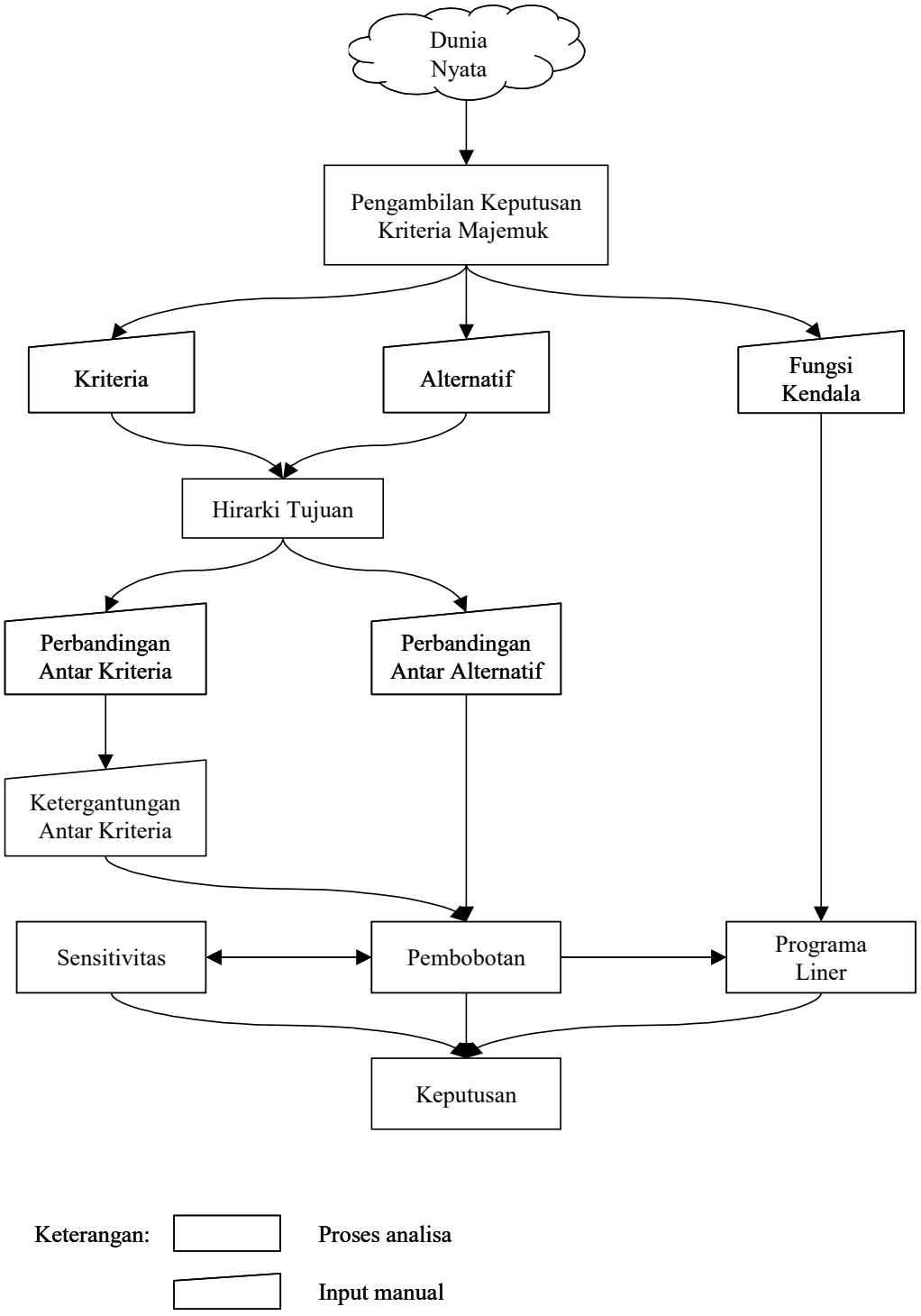
Dalam analisis sistem, pendekatan *top-down* ini disebut juga dengan istilah *decision analysis* karena yang menjadi tekanan adalah informasi yang dibutuhkan untuk pengambilan keputusan oleh manajemen terlebih dahulu, kemudian data yang perlu diolah didefinisikan menyusul mengikuti informasi yang dibutuhkan.

Tiap subsistem memiliki sendiri input dan output yang berdiri sendiri, dan responnya dapat dijelaskan dengan melihat hubungan yang ada dalam subsistem tersebut. Interaksi-interaksi yang terjadi di dalam suatu sistem ada karena dimungkinkan oleh fakta bahwa output dari beberapa subsistem menjadi input bagi subsistem-subsistem yang lainnya.

Desain sistem yang dirumuskan dalam penelitian ini dirancang dengan tujuan untuk memudahkan komunikasi dengan pemakai, penggunaan memori secara optimum, dan kemudahan dalam memperoleh solusi hasil perhitungan algoritma model atas suatu permasalahan pengambilan keputusan kriteria majemuk.

Berdasarkan kebutuhannya, analisis sistem untuk perancangan RATIONAL merekomendasikan dua subsistem operasi dari program utama, yang terdiri dari:

1 Subsistem pengambilan keputusan individual, yang memuat tentang:
   a Pembuatan hirarki:
      (1) hubungan antar kriteria
      (2) hubungan antara alternatif dan kriteria
   b Perbandingan antar elemen:
      (1) tingkat kepentingan antar kriteria,
      (2) tingkat kepentingan antar alternatif,
   c tingkat ketergantungan (interdependensi) antar kriteria,
   d analisis pembobotan,
   e analisis sensitivitas,
   f programa linier.



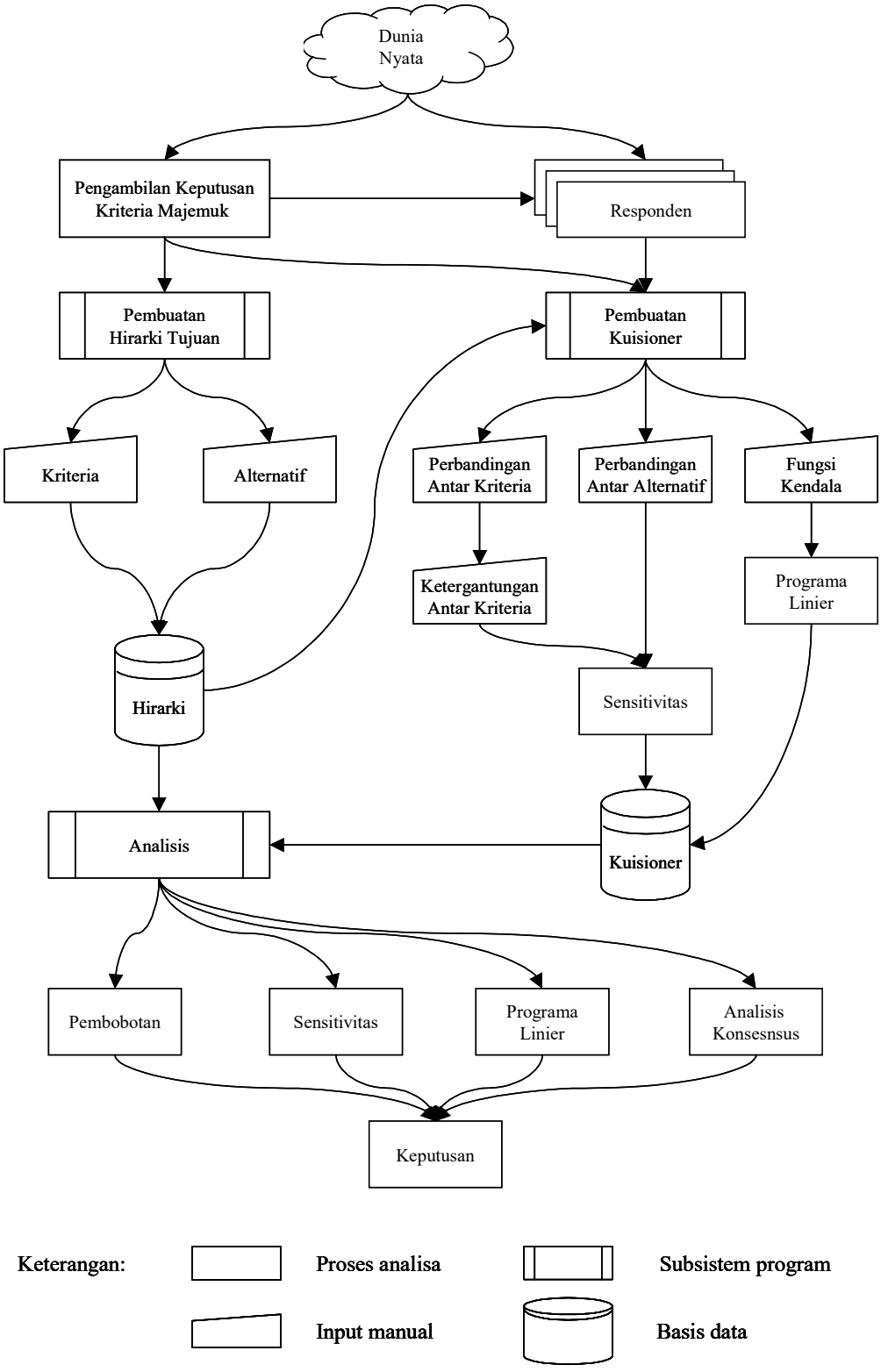
Gambar V.5 Kerangka Rancangan Sistem Pengambilan Keputusan Individual

2 Subsistem pengambilan keputusan kelompok, yang memuat subsistem-subsistem:

a Pembuatan hirarki, yang memuat tentang:
   (1) hubungan antar kriteria
   (2) hubungan antara alternatif dan kriteria

b Kuisioner, yang terdiri dari isian data dari responden yang memuat tentang:
   (1) tingkat kepentingan antar kriteria,
   (2) tingkat kepentingan antar alternatif,
   (3) tingkat ketergantungan (interdependensi) antar kriteria,
   (4) kendala implementasi alternatif;

c Analisis data, yang memuat tentang analisis data hasil dari kuisioner, yang terdiri dari
   (1) analisis pembobotan,
   (2) analisis sensitivitas,
   (3) analisis perbandingan antar responden
   (4) programa linier.



Gambar V.6 Kerangka Rancangan Sistem Pengambilan Keputusan Kelompok

2.1 Desain Input

Desain input dalam penelitian ini merujuk pada kebutuhan model yang kemudian dikomunikasikan melalui sebuah menu dialog. Input merupakan awal dimulainya proses transformasi dari data menjadi informasi, yang kemudian diproses sebagai referensi dalam pembuatan keputusan. Data input yang digunakan dalam RATIONAL merupakan masukan berdasarkan investigasi yang terjadi dalam dunia nyata yang mampu ditangkap oleh pengambil keputusan. Hasil dari RATIONAL, tidak lepas dari derajat validitas data yang dimasukkan ke dalam sistem, "*garbage in garbage out*".

Berdasarkan bentuk model PKKM yang dikembangkan, maka dapat diidentifikasikan bahwa kebutuhan input yang perlu diintegrasikan ke dalam sistem adalah:

1 Pengambilan keputusan individual, terdiri dari:
   - a Tujuan (solusi yang diinginkan)
   - b Kriteria-kriteria yang dipertimbangkan dalam pemilihan alternatif
   - c Alternatif-alternatif yang mungkin
   - d Susunan hirarki tujuan
   - e Preferensi antar elemen (kriteria-kriteria dan alternatif-alternatif)
   - f Kecenderungan keterkaitan antar elemen
   - g Kendala teknologi yang diperkirakan muncul dengan implementasi suatu alternatif

2 Pengambilan keputusan kelompok, terdiri dari:
   - a Pembuatan hirarki
     - (1) Tujuan (solusi yang diinginkan)
     - (2) Kriteria-kriteria yang dipertimbangkan dalam pemilihan alternatif
     - (3) Alternatif-alternatif yang mungkin
     - (4) Susunan hirarki tujuan

b Pengisian Kuisioner

(1) Data responden

- Nama
- Pekerjaan
- Alamat
- E-mail
- Keterangan

(2) Preferensi antar elemen (kriteria-kriteria dan alternatif-alternatif)

(3) Kecenderungan keterkaitan antar elemen

(4) Kendala teknologi yang diperkirakan muncul dengan implementasi suatu alternatif

c Analisa

(1) Pemilihan hirarki tujuan

(2) Pemilihan data hasil kuisioner

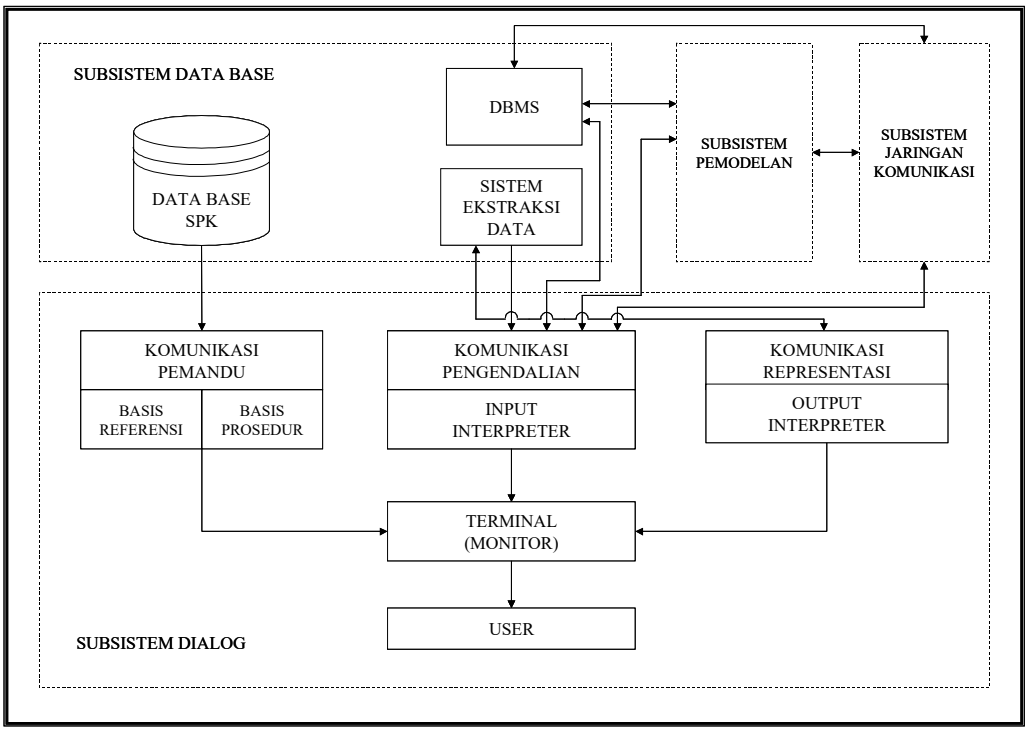
Model input data dalam SPK tergantung dari proses input yang dibutuhkan, dalam sistem ini digunakan dua tahapan utama, yaitu penangkapan data (*data capture*) dan pemasukan data (*data entry*). Penangkapan data merupakan proses mencatat kejadian nyata dalam sebuah kuisioner elektronik (pada model pengambilan keputusan kelompok) yang diisi oleh orang yang dinyatakan sebagai ahli dalam bidang yang diteliti. Pemasukkan data merupakan suatu proses memasukkan data oleh user ke dalam sistem (terkomputerisasi).

2.2 Desain Dialog

Desain dialog yang digunakan dalam RATIONAL merupakan rancang bangun dari komunikasi antara pemakai sistem dengan komputer. Komunikasi ini berupa proses memasukkan data ke sistem dan informasi dari sistem. Desain dialog dalam sistem berdasarkan spesifikasi model PKKM yang ditampilkan melalui menu interaktif, yang memiliki kapabilitas pertanyaan untuk

memperoleh informasi sesuai dengan kebutuhan, dengan pendekatan ini diharapkan sistem memenuhi kriteria *easy to use*.

Fungsi dan fleksibilitas RATIONAL dibangun atas pertimbangan kemudahan interaksi antara sistem dengan pemakainya (pengambil keputusan). Interaksi ini berlangsung dalam subsistem yang terdiri dari perangkat lunak, terminal (*console monitor*) dan pemakai, yang membentuk suatu sistem dialog.



Gambar V.7 Rancangan Subsistem Dialog

Dialog antara pemakai dengan sistem dilakukan melalui *bahasa komunikasi*. Bahasa komunikasi yang diterapkan dalam dialog *system-user* pada RATIONAL, dikategorikan dalam tiga jenis, yaitu:

a Komunikasi antara pemakai dengan sistem yang dalam aktivitas pengendalian operasinya dilakukan melalui serangkaian dialog mengenai kegiatan yang dilakukan oleh pemakai untuk berkomunikasi dengan

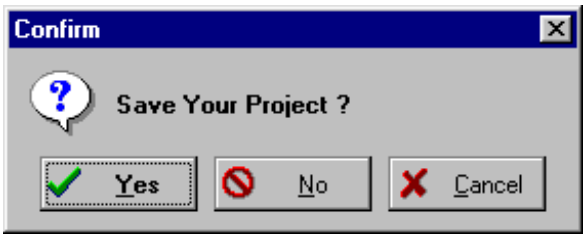
sistem, tipe/gaya dialog serta fasilitas yang bisa dipakai untuk menginterupsi proses yang sedang berlangsung.

Pada RATIONAL, kebanyakan bentuk komunikasi antara pemakai dengan sistem dalam aktivitas pengendalian operasinya dilakukan dengan dialog menu dan tanya-jawab. Dialog yang berdasarkan tanggapan/jawaban yang diberikan oleh pemakai terhadap pertanyaan yang diajukan sistem ini, dapat dibedakan sebagai berikut:

- *Pilihan*, dimana sistem mengajukan beberapa alternatif pilihan kepada pengambil keputusan. Dialog ini diaplikasikan dalam program menu pengendali, pemilihan tahapan atau algoritma dalam pemodelan, pemilihan item data atau kolom data yang akan diolah, dan sebagainya.

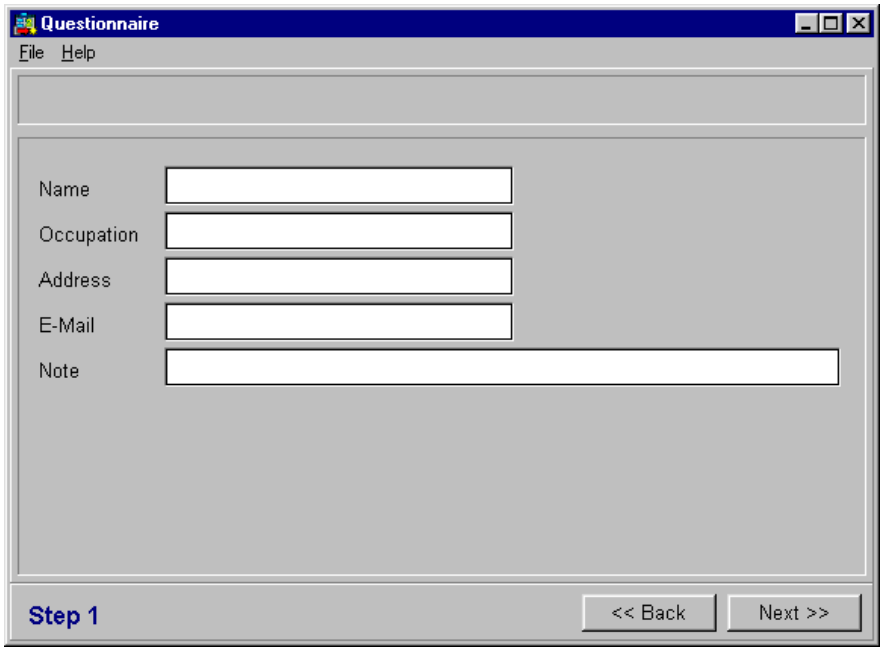
Gambar V.8 Contoh Dialog Menu Pilihan pada RATIONAL

- *Persetujuan*, dimana pernyataan yang diajukan oleh sistem guna mendapatkan persetujuan dari pemakai. Bentuk ini diaplikasikan pada penentuan pilihan diantara dua alternatif dan umumnya pada operasi-operasi tambahan, seperti penulisan laporan ke printer, analisis tambahan dan sebagainya.



Gambar V.9 Contoh Dialog Menu Persetujuan pada RATIONAL

- *Isian*, dimana pertanyaan-pertanyaan yang diajukan harus dijawab oleh pemakai dengan mengisi bagian kosong dengan jawaban yang dianggap tepat. Jenis tanya-jawab ini diaplikasikan pada pemasukan parameter-parameter yang berkaitan dengan aktivitas pemodelan.



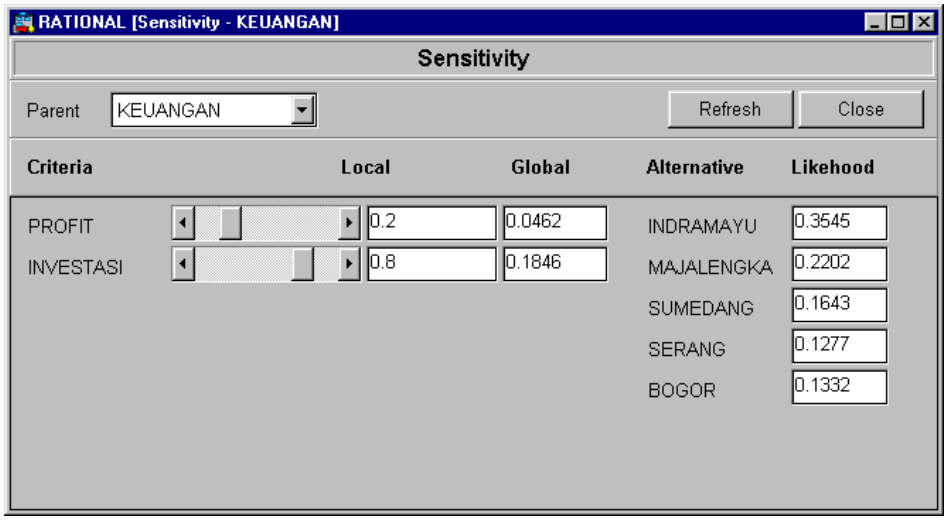
Gambar V.10 Contoh Dialog Menu Isian pada RATIONAL

Dalam RATIONAL, komponen dialog tanya-jawab dilengkapi dengan fasilitas yang berfungsi mengkoreksi jawaban atau tanggapan yang salah, dengan cara memeriksa apakah jawaban tersebut berada dalam batas-

batas yang ditoleransi, atau apakah jawaban tersebut termasuk salah satu alternatif jawaban yang ada.

b Komunikasi Peraga atau Representasi

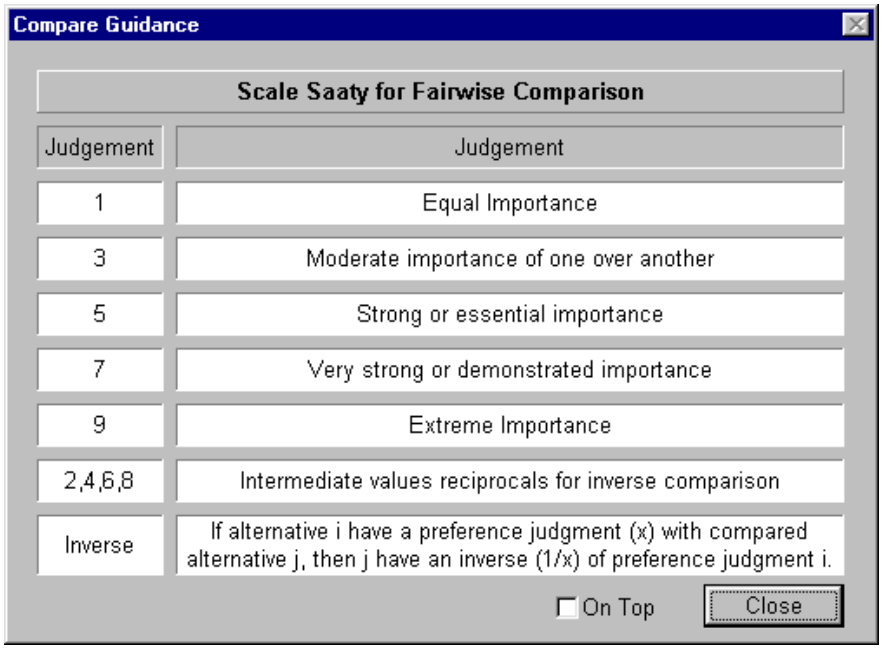
Selama proses, sistem memberikan informasi kepada pemakainya, berupa *feed-back* terhadap instruksi-instruksi yang diberikan oleh pemakainya, informasi tentang status proses yang sedang berlangsung, informasi dalam bentuk laporan hasil proses, atau representasi model yang dibuat melalui komponen pemodelan.



Gambar V.11 Contoh Menu Dialog Komunikasi Peraga/Representasi pada RATIONAL

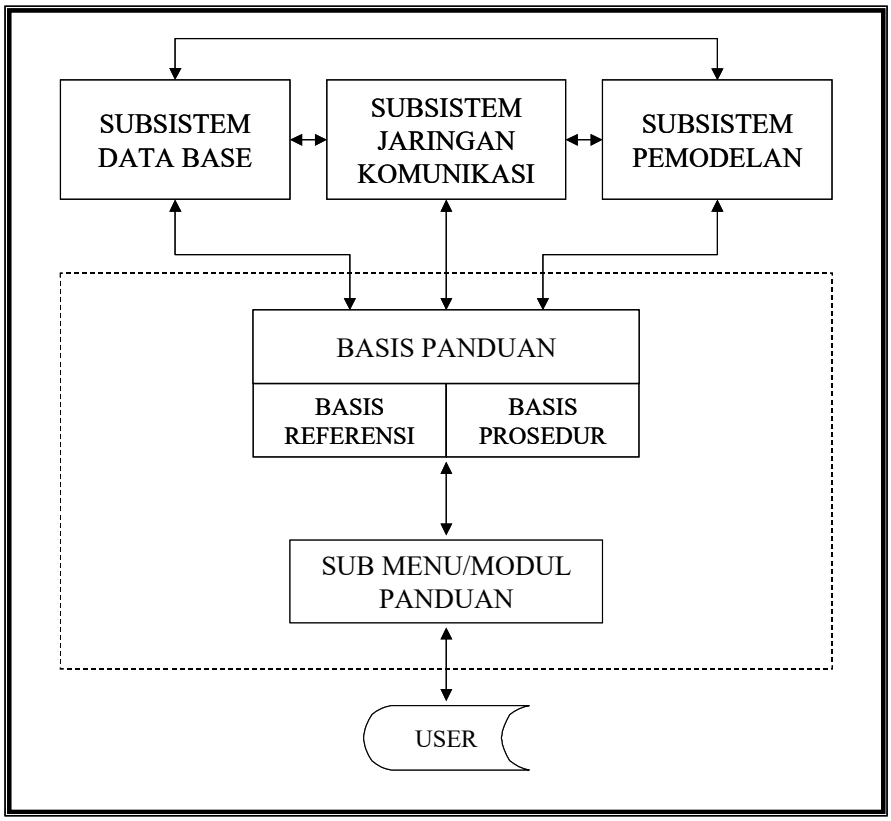
c Komunikasi Pemandu

Bahasa jenis ini merupakan komunikasi yang berkaitan dengan hal-hal yang mesti dipahami oleh pemakai guna mengaktifkan sistem secara efektif, untuk itu pemakai harus mempunyai pengetahuan mengenai struktur sistem dan prosedur umum untuk mengoperasikannya.



Gambar V.12 Contoh Menu Dialog Komunikasi Pemandu pada RATIONAL

Pada RATIONAL, pengetahuan yang dimaksud di atas, bisa disajikan melalui menu basis referensi yang diproses secara interaktif dalam sub-paket panduan.



Gambar V.13 Konfigurasi Basis Panduan

Secara operasional, kebijakan-kebijakan yang menjadi petunjuk dasar bagi pengambilan keputusan harus dijabarkan menjadi prosedur-prosedur yang bisa memandu tahapan-tahapan pengambilan keputusan. Prosedur-prosedur ini mencerminkan model-model keputusan dan jenis-jenis informasi yang dipakai dalam proses pengambilan keputusan tersebut.

Pada RATIONAL, komponen yang menangani fungsi ini adalah komponen Basis Prosedur, yang juga menjadi bagian komponen basis panduan. Sebagai pemandu aktivitas pengambilan keputusan, Komponen Basis Prosedur ini setiap saat diperlukan.

2.3 Desain Model

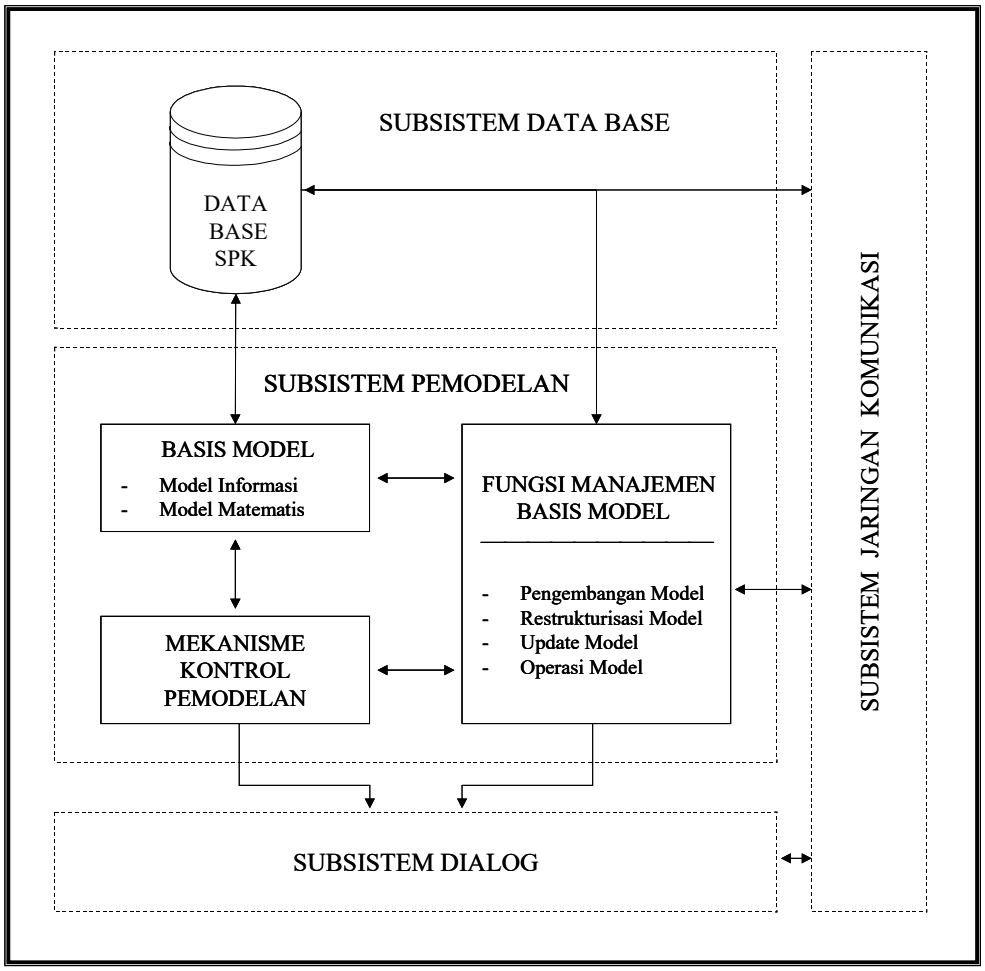
Secara ideal, komponen pemodelan harus menunjang setiap aktivitas pengambilan keputusan yang meliputi: analisis sistem permasalahan, proyeksi situasi masa depan, perancangan alternatif, pembandingan atau pemilihan alternatif, optimasi dan simulasi melalui penerapan model-model yang relevan.

Model-model yang digunakan dalam proses pengambilan keputusan pada RATIONAL dapat dikategorikan dalam dua jenis, yaitu:

a *Model Matematik*, yaitu model yang mempresentasikan sistem secara simbolik dengan menggunakan rumus-rumus atau besaran-besaran abstrak. Model ini selanjutnya bisa dijabarkan ke dalam operasi-operasi matriks, algoritma iteratif dan model-model keputusan matematis lainnya.

b *Model Informasi*, yaitu model yang mempresentasikan sistem dalam format grafik atau tabel. Secara umum, model ini bisa dibagi atas:

- *Penjelasan objek*, mendeskripsikan suatu objek secara terperinci, berupa tabel, daftar, dan sebagainya.
- *Penjelasan hubungan,* menunjukkan hubungan antar objek. Pada model ini, representasi hubungan ditampilkan dalam bentuk grafik
- *Penjelasan operasi*, menunjukkan urutan tugas atau proses yang dilakukan oleh suatu objek atau sekelompok objek.



Gambar V.14 Konfigurasi Subsistem Pemodelan

Model-model pada RATIONAL dirancang sedemikian rupa sehingga satu sama lain bersifat independen. Dengan demikian untuk setiap basis model yang dipilih, pengambil keputusan dapat menentukan sendiri parameter-parameter model yang digunakan, tahapan atau algoritma yang akan dijalani oleh proses pemodelan yang bersangkutan melalui komponen dialog. Alternatif ini dipilih guna mendapatkan sistem pendukung kegiatan pemodelan yang fleksibel.

Kerangka utama yang digunakan dalam desain model yang digunakan dalam penelitian ini, adalah penggunaan model *physical system*. *Physical system* ini

direpresentasikan melalui sebuah bagan alir sistem. Gambaran bagan alir model disajikan Gambar V.5 dan Gambar V.6.

Berdasarkan bagan alir sistem ini teridentifikasikan metode pengolahan data. Adapun prosedur pengolahan data dalam sistem ini mengikuti prosedur sebagai berikut:

1 Pengambilan keputusan individual
   a Investigasi permasalahan dunia nyata yang mempersyaratkan penyelesaiannya melalui instrumen PKKM.
   b Penelusuran kriteria-kriteria yang dijadikan bahan pertimbangan dalam pemecahan masalah dari dunia nyata.
   c Penelusuran alternatif-alternatif yang mungkin direalisasikan sebagai pemecahan masalah.
   d Penyusunan hirarki tujuan.
   e Pengisian data preferensi perbandingan antar elemen
   f Pengisian data tingkat interdependensi elemen
   g Pengisian data kendala implementasi alternatif
   h Pengolahan data bobot elemen, tingkat interdependensi elemen, kecenderungan alternatif dan idealisasi implementasi alternatif (programa linier berdasarkan kendala teknologi yang ditemui dalam dunia nyata).
   i Analisis sensitivitas
   j Pembuatan laporan.

2 Pengambilan keputusan kelompok

a Pembuatan hirarki

(1) Investigasi permasalahan dunia nyata yang mempersyaratkan penyelesaiannya melalui instrumen PKKM.

(2) Penelusuran kriteria-kriteria yang dijadikan bahan pertimbangan dalam pemecahan masalah dari dunia nyata.

(3) Penelusuran alternatif-alternatif yang mungkin direalisasikan sebagai pemecahan masalah.

(4) Penyusunan hirarki tujuan.

b Pencarian data perbandingan

(1) Penyusunan kuisioner sebagai dokumen dasar input sistem.

(2) Penetapan responden yang dianggap mengerti dan mengenal permasalahan tersebut, pada tahap ini dicari data tentang:

(a) Preferensi perbandingan antar elemen

(b) Tingkat interdependensi elemen

(c) Kendala implementasi alternatif

c Pengolahan data

(1) Analisis konsensus

(2) Pengolahan data bobot elemen, berdasarkan tingkat kepentingan antar elemen dan interdependensi elemen,

(3) Kecenderungan alternatif

(4) Analisis sensitivitas

(5) Idealisasi implementasi alternatif

(6) Pembuatan laporan

2.4 Desain Basis Data

Basis data (*data base*) merupakan sekumpulan data yang tersimpan dan terstruktur rapi dalam sistem. Mengingat SPKKM yang digunakan untuk model ini ukurannya kecil, maka model yang digunakannya adalah model relasional. Basis data ini menyimpan bentuk-bentuk data hirarki tujuan, preferensi, interdependensi, dan fungsi kendala.

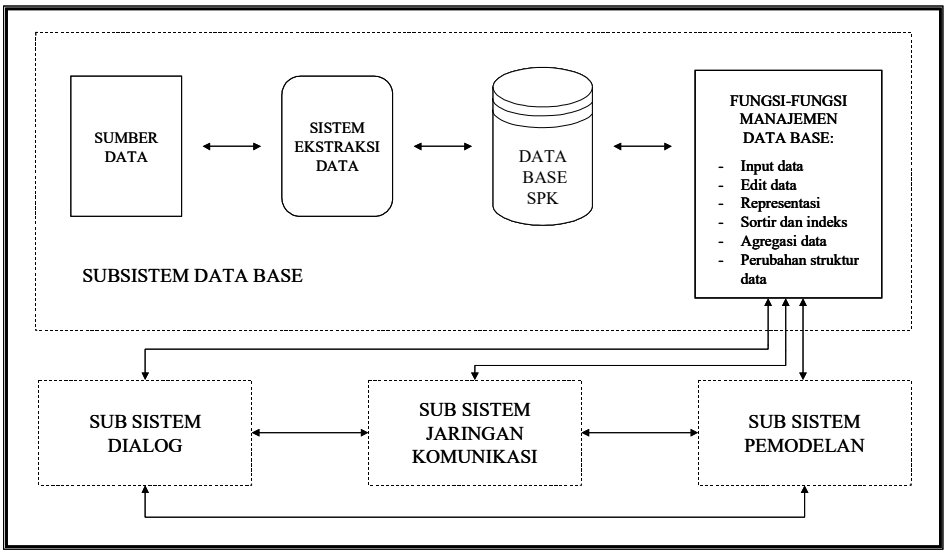
Setiap data dicatat sebagai suatu struktur basis data (*record*), yang masing-masing terdiri dari beberapa struktur kolom data (*field*). Jumlah dan tipe (numerik/alphabet) field suatu file dijabarkan sesuai dengan kebutuhan, dengan menggunakan fasilitas fungsi Subsistem Manajemen Basis data.

Konfigurasi Subsistem Basis data dibagi atas dua prosedur, yaitu subsistem basis data dan subsistem ekstraksi data, dengan uraian sebagai berikut:

a Subsistem Manajemen Basis data

Subsistem Manajemen Basis data berfungsi sebagai pengelola basis data. Fungsi-fungsi subsistem ini meliputi:

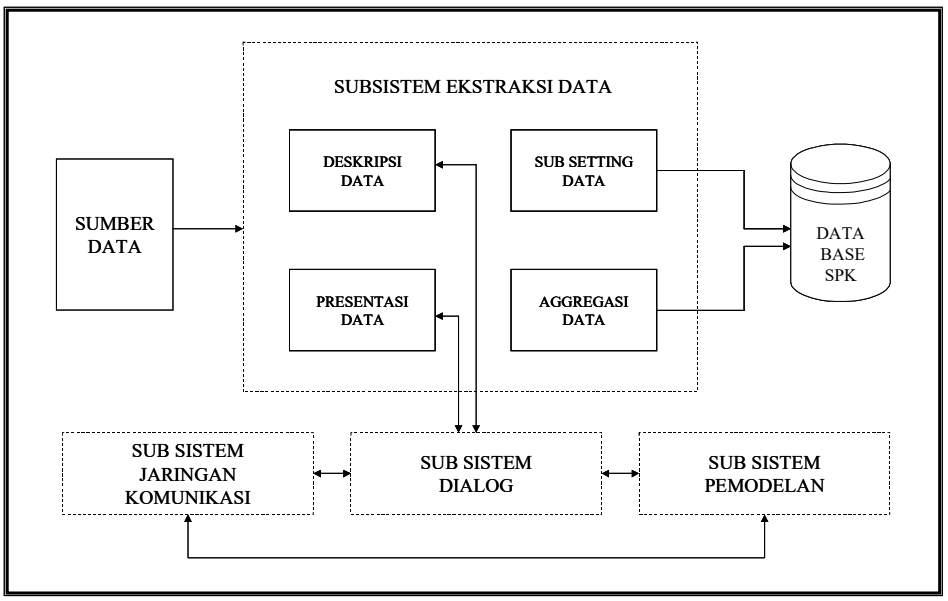
- pemasukan data
- penambahan data
- perubahan struktur berkas data
- modifikasi data
- penghapusan data
- penjabaran (*list*) data
- sortir (urut) data
- duplikasi berkas data
- integrasi/agregasi berkas data



Gambar V.15 Konfigurasi Subsistem Manajemen Basis Data

b Subsistem Ekstraksi Data

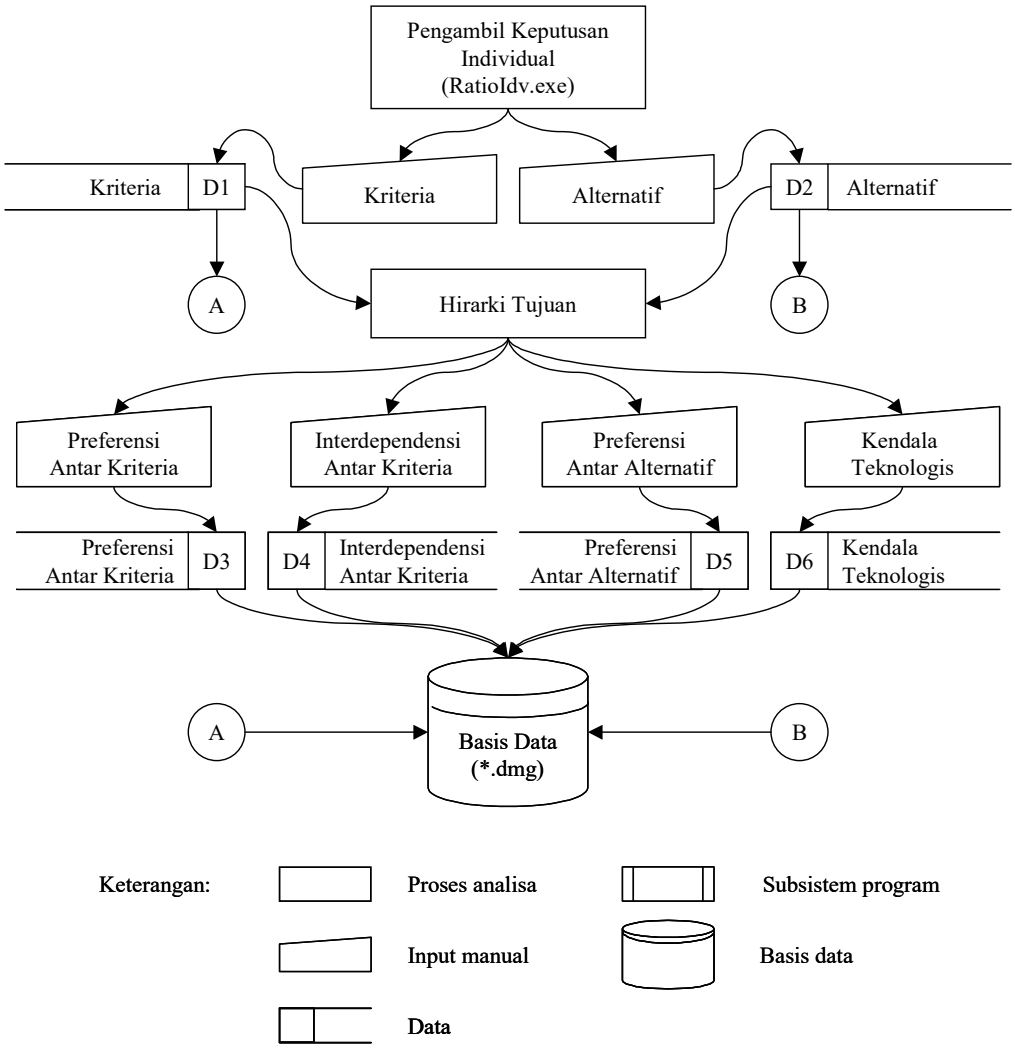
Dengan adanya komponen ekstraksi data, file-file dari basis data sumber dapat diorganisasikan guna menunjang analisis dan presentasi data yang dibutuhkan SPKKM.



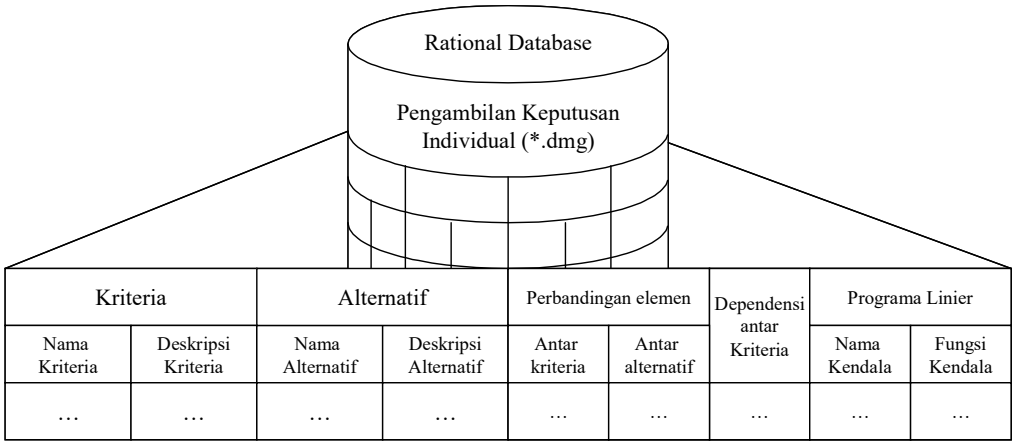
Gambar V.16 Konfigurasi Subsistem Ekstraksi Data

Melalui komponen dialog, pemakai/user dapat memberikan instruksi pada komponen ekstraksi data untuk memilih berkas data, dan menentukan lingkungan berkas data yang akan diolah.

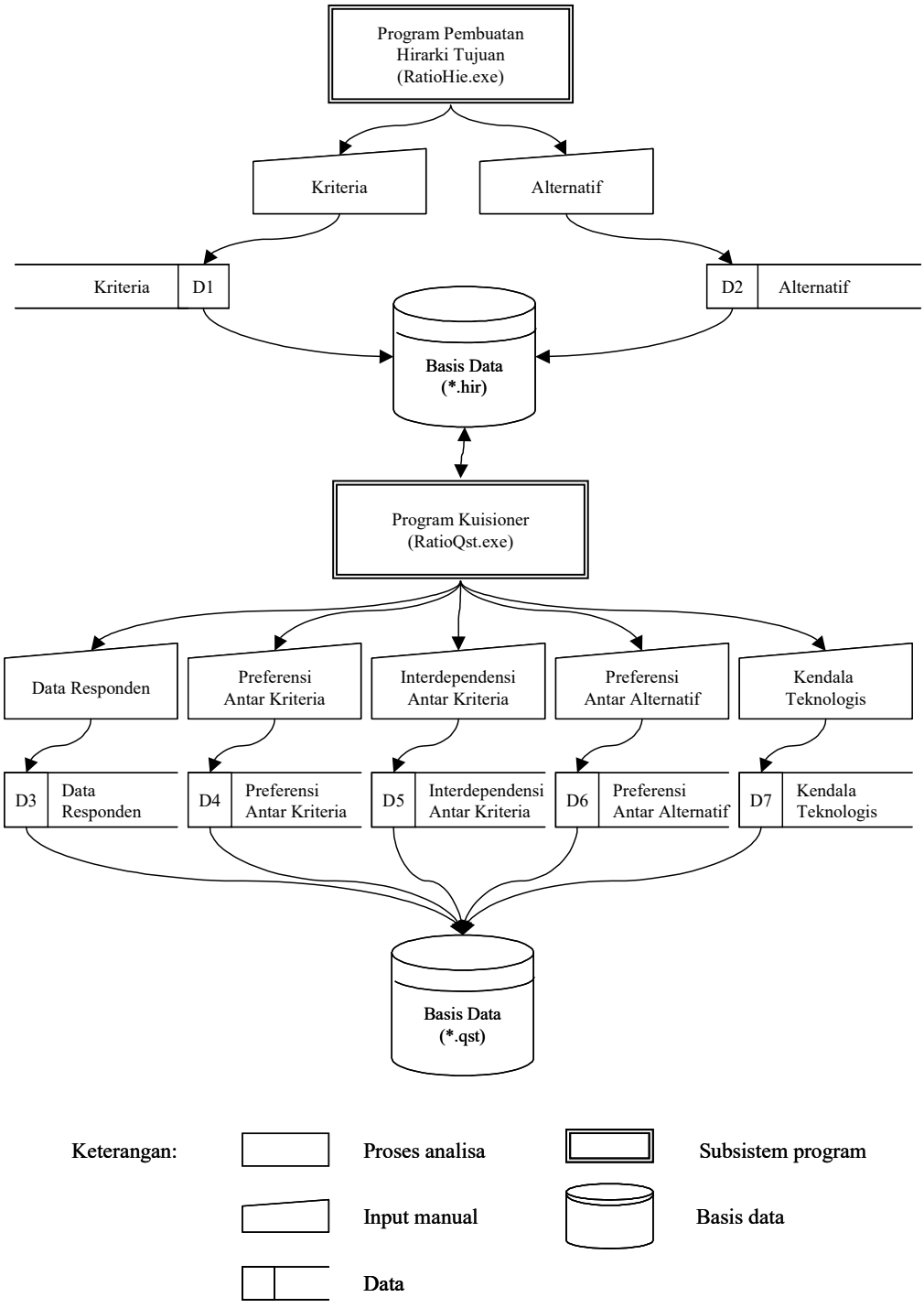
Kebutuhan basis data untuk pengambilan keputusan individual disajikan pada Gambar V.17 dan Gambar V.18, sedangkan untuk pengambilan keputusan kelompok disajikan pada Gambar V.19, Gambar V.20, dan Gambar V.21.



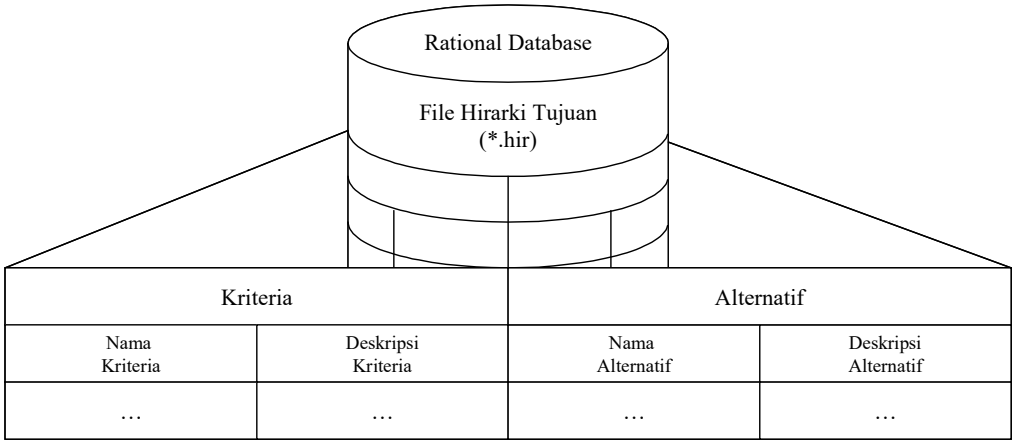
Gambar V.17 Diagram Aliran Data untuk Pengambilan Keputusan Individual



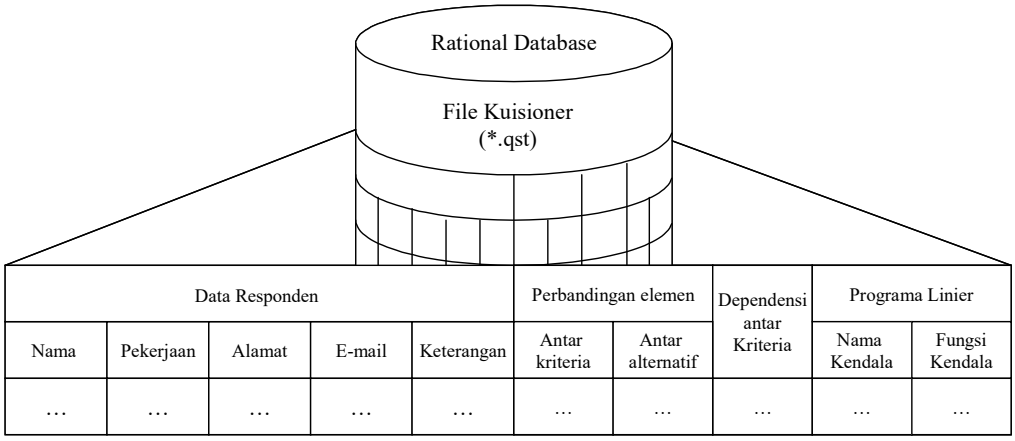
Gambar V.18 Basis Data untuk Pengambilan Keputusan Individual



Gambar V.19 Diagram Aliran Data untuk Pengambilan Keputusan Kelompok



Gambar V.20 Basis Data untuk Program Pembuatan Hirarki



Gambar V.21 Basis Data untuk Program Kuisioner

Elemen-elemen data dalam suatu file basis data harus dapat digunakan untuk pemeliharaan data dalam sistem. Adapun bentuk desain basis data sistem disyaratkan harus dapat membuat file baru, menghapus data lama, menyisipkan data, atau mengubah data.

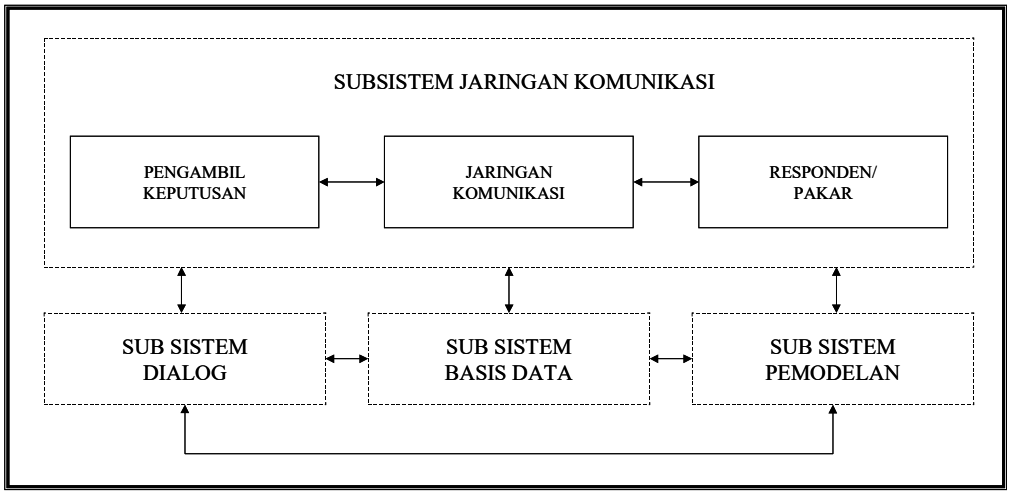
2.5 Desain Jaringan Komunikasi

RATIONAL merupakan SPK Kelompok. Karena sebagai SPK Kelompok, maka RATIONAL dipersyaratkan untuk menjadi sistem yang mendukung interaksi individu dan kelompok pada proses pengambilan keputusan. Sistem ini memiliki tiga fungsi utama, yaitu penyampaian/distribusi informasi, pengendalian, dan user-interface.

Pendistribusian informasi memungkinkan masing-masing individu anggota kelompok untuk melakukan diskusi. Fungsi ini mempersyaratkan spesifikasi model data yang konsisten (ekivalen). Pengendalian dilakukan oleh pengambil keputusan dengan mengidentifikasi kesesuaian data dan pengendalian jalannya diskusi. User-interface menangani komunikasi antara individu dan sistem. Pada fungsi user-interface diatur pengelolaan yang bersifat privat dan kelompok. Fungsi user-interface mempersyaratkan sistem untuk dapat menerima informasi dan penanganan sistem pada skala kelompok dan individu.

Model konsensus yang dikembangkan dari model menggunakan kaidah-kaidah metoda Delphi. Metoda Delphi tidak mensyaratkan individu anggota kelompok untuk dapat bekerja pada waktu dan tempat yang sama. Sehingga sistem harus mampu mendistribusikan data yang dapat dikirim ke suatu tempat dan disimpan dalam ruang waktu yang fleksibel, artinya sistem harus menyediakan model penyimpanan data yang dapat diakses dari tempat yang mungkin berbeda dan waktu yang mungkin berbeda.

Desain jaringan komunikasi yang dilakukan dalam SPK dilakukan berdasarkan kebutuhan sistem untuk mendistribusikan dan menerima data dari sekelompok pakar/responden. Jaringan komunikasi pada SPK ini memanfaatkan teknologi yang telah ada, yaitu jaringan internet.



Gambar V.22 Konfigurasi Subsistem Jaringan Komunikasi

Bentuk komunikasi yang dibutuhkan sistem adalah:

a Distribusi paket software RATIONAL oleh pengambil keputusan kepada seluruh responden;

b Distribusi masalah berupa data (*.hir) oleh pengambil keputusan kepada seluruh responden;

c Pengembalian data (*.rsp) dari responden kepada pengambil keputusan.

d Pesan berupa masalah, argumentasi, dan penjelasan dalam bentuk teks.

2.6 Desain Teknologi

Analisis desain teknologi yang dilakukan dalam SPKKM dilakukan berdasarkan identifikasi jenis dari teknologi yang dibutuhkan dan jumlahnya yang diperlukan oleh paket aplikasi RATIONAL.

Untuk teknologi perangkat keras, penentuan jenis teknologi dilakukan berdasarkan identifikasi peralatan yang digunakan di masing-masing proses dalam RATIONAL, sedangkan perangkat lunak pemilihannya dilakukan berdasarkan kebutuhan sistem dibandingkan dengan keandalan perangkat lunak dalam menjalankan sasaran implementasi RATIONAL.

Desain teknologi yang disusun dalam sistem ini digunakan dalam tujuan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan output, dan membantu pengendalian dari sistem secara keseluruhan.

1 Alat Masukan

Alat masukan (*input device*/*input unit*/*input equipment*) merupakan perangkat yang digunakan untuk memberikan masukan bagi sistem. Input yang digunakan dalam sistem ini terdiri dari input langsung yang berupa *keyboard*, dan *pointing device* berupa *mouse*, dan alat input tak langsung berupa disk magnetik (*magnetic disk*).

2 Alat Pemroses

Alat pemroses merupakan perangkat yang digunakan untuk menjalankan instruksi-instruksi program untuk mengolah data yang menjadi masukan bagi sistem. Alat pemroses yang digunakan dalam sistem ini adalah *central processor* atau CPU (*Central Processing Unit*), dan *main memory*.

3 Alat Output

Output yang dihasilkan oleh sistem dapat digolongkan ke dalam dua bentuk, yaitu tulisan dan image. Kebutuhan atas sistem untuk menampilkan bentuk output ini disajikan melalui alat-alat *hard copy device* berupa printer, *soft copy device* berupa layar monitor, *modem*, jaringan telepon dan *drive device* berupa penggerak disk magnetik.

4 Simpanan Luar

Main memory di dalam alat pemrosesan merupakan simpanan yang kapasitasnya tidak begitu besar, dan umumnya bersifat *volatile*, yaitu informasi yang dikandungnya akan hilang bila aliran listrik putus. Untuk itu, diperlukan suatu simpanan yang mempunyai kapasitas besar dan bersifan *non-valatile* untuk menyimpan data dan program dalam kurun waktu tertentu. Simpanan luar ini memiliki sifat sebagai *external memory* (simpanan luar).

5 Perangkat Lunak

Teknologi perangkat keras akan berfungsi manakala intruksi-instruksi yang diberikan kepadanya dapat dijalankan, untuk itu digunakan teknologi perangkat lunak sebagai fasilitas untuk menterjemahkan instruksi menjadi aksi dalam sistem. Sistem ini menggunakan dua kategori perangkat lunak, yaitu perangkat lunak sistem operasi (*operating sytem*), yaitu program yang digunakan untuk mengendalikan dan mengkoordinasikan kegiatan dari sistem komputer, dan perangkat lunak bahasa (*language software*), yaitu program yang digunakan untuk menterjemahkan instruksi-instruksi yang ditulis dalam bahasa pemrograman ke dalam bahasa mesin agar dapat dimengerti oleh komputer. *Operating system* yang digunakan dalam RATIONAL adalah Windows '95 dan *language software* yang digunakan adalah bahasa pemrograman *Borland Delphi* versi 3.0. Untuk jaringan komunikasi digunakan paket *Netscape* atau *Microsoft Internet Explorer*.

2.6 Desain Kontrol

RATIONAL merupakan sebuah sistem yang terbuka (*open system*), sehingga dalam menjalankan aktivitasnya memerlukan penanganan pengendalian. Desain kontrol merupakan desain atas instrumen-instrumen sistem untuk mengendalikan aplikasi sistem secara optimal.

Pengendalian yang dilibatkan dalam RATIONAL terdiri atas pengendalian masukan, pengendalian pengolahan, dan pengendalian keluaran, serta pengendalian keamanan data. Pengendalian diterapkan selama proses pengolahan data berlangsung.

1 Pengendalian masukan

Pengendalian masukan mempunyai tujuan untuk meyakinkan bahwa data masukan yang valid telah lengkap, terkumpul semuanya serta bebas dari kesalahan sebelum dilakukan proses pengolahan. Pengendalian input ini merupakan pengendalian yang penting, karena input yang salah akan mengakibatkan output yang salah.

2 Pengendalian pengolahan

Pengendalian pengolahan bertujuan untuk mencegah kesalahan yang terjadi selama proses pengolahan data. Kesalahan pengolahan dapat terjadi karena program aplikasi yang digunakan untuk mengolah data mengandung kesalahan. Untuk menditeksi kesalahan-kesalahan yang mungkin terjadi, maka pada tahap ini dapat dilakukan beberapa pengendalian berupa pengujian-pengujian. Pada tahap pengolahan data, dapat terjadi kesalahan-kesalahan yang umumnya disebabkan oleh kesalahan dalam program, seperti misalnya *overflow*, kesalahan logika program, logika program yang tidak lengkap, penangangan pembulatan yang salah, dan kesalahan pembacaan data di file acuan. Kesalahan-kesalahan yang terjadi selama tahap pengolahan dapat dikendalikan dengan menguji proses dari program. Program bahasa komputer umumnya dibuat sedemikian rupa sehingga kesalahan-kesalahan yang terjadi dalam pengolahan data dapat dideteksi. Pengujian yang dilakukan dalam pengendalian pengolahan antara lain:

- *Reference File Check*

  Kesalahan penggunaan data yang diambil dari file acuan (*reference file*) dapat dideteksi dengan cara mencetak isi file acuan yang digunakan setelah proses pengolahan dan kemudian dibandingkan hasil yang diinginkan.

- *Limit and Reasonabel Check*

  Pengendalian dilakukan dengan pegujian terhadap batas limit dan batas kewajaran suatu nilai hasil pengolahan.

3 Pengendalian keluaran

Keluaran merupakan produk dari pengolahan data yang disajikan pada dua bentuk utama, yaitu *hard copy* dan *soft copy*. Pengendalian ini digunakan dalam penyusunan laporan yang baik guna menghindari salah interpretasi output oleh pengguna sistem.

4 Pengendalian keamanan data

Pengendalian keamanan data digunakan untuk menjaga data dari kerusakan dan hilangnya data. Prosedur pengendalian keamanan data yang digunakan dalam RATIONAL adalah dimungkinkannya pembuatan data *back up* dan *recovery*.

3 Perancangan Program Aplikasi

Tahap perancangan program aplikasi merupakan tahap pembuatan dan penggunaan operasi sistem. Pada tahap ini, juga termasuk menulis kode program pembuatan program aplikasi RATIONAL. Perancangan program aplikasi dilakukan dengan meninjau hubungan diantara komponen-komponen kerja SPK dan mempertimbangkan pemilihan arsitektur SPK. Perancangan program aplikasi merupakan integrasi konfigurasi model dengan arsitektur sistem.

Untuk menghubungkan antara satu komponen dengan komponen lain, dan memindahkan mekanisme pengendalian sistem dari suatu komponen ke komponen lain digunakan suatu modul komponen pengintegrasi atau *component interface*. Penghubung komponen dalam hal ini berupa sub-program (*subroutine*) yang ada pada setiap komponen sistem, berfungsi untuk membaca dan menulis parameter sistem dari memori. Parameter sistem ini juga mengidentifikasikan fungsi-fungsi pengendalian program dan alokasi memori.

3.1 Pemrograman Paket Aplikasi

Pemrograman paket aplikasi pada dasarnya merupakan transformasi dari bahasa umum dan atau bahasa formulasi matematis menjadi kode-kode atau perintah-perintah yang dipahami oleh komputer. Pembuatan program aplikasi RATIONAL untuk penelitian ini digunakan bahasa pemrograman Pascal Delphi.

Pascal Delphi adalah sebuah aplikasi untuk pengembangan yang memanfaatkan keistimewaan konsep-konsep antar muka grafis dalam Microsoft Windows. Microsoft Windows memiliki antar muka grafis (*Graphical User Interface*) yang menjadi lingkungan kerja standar bagi pemakai/pemrogram. Sistem operasi tersebut menjalankan program-program 32-bit. Delphi dapat mengkompilasi program 32-bit untuk sistem operasi yang baru, dan dapat menghasilkan program yang berakselerasi sangat cepat.

Program aplikasi yang dihasilkan terdiri dari lima program yang bereksistensi *.exe, dan satu program berinisial *.ini, yaitu:

a Rational.exe sebagai program utama

b RatioIdv.exe merupakan program aplikasi untuk pengambilan keputusan individual

c RatioHie.exe sebagai program aplikasi untuk membuat hirarki tujuan untuk pengambilan keputusan kelompok

d RatioQst.exe merupakan program aplikasi untuk melakukan pengisian data dari responden untuk pengambilan keputusan kelompok

e RatioGrp.exe merupakan program aplikasi untuk melakukan pengolahan data pada pengambilan keputusan kelompok

f Rational.ini sebagai penghubung antara program utama dengan program aplikasi

3.2 Pengujian Program Paket Aplikasi

Pengujian program dilakukan dengan tujuan agar program aplikasi bebas dari kesalahan-kesalahan. Pengujian RATIONAL dilakukan untuk setiap modul dan dilanjutkan dengan pengujian untuk semua modul. Kesalahan program yang diuji diklasifikasikan dalam tiga bentuk kesalahan, yaitu:

1 Kesalahan bahasa (*language errors*), atau kesalahan penulisan (*sytax errors*) atau kesalahan tata bahasa (*grammatical errors*). Kesalahan ini merupakan kesalahan di dalam penulisan *source program* yang tidak sesuai dengan yang disyaratkan.

2 Kesalahan sewaktu proses (*run-time errors*), yaitu kesalahan yang terjadi pada waktu *executable program* dijalankan. Kesalahan ini akan menyebabkan proses program berhenti sebelum selesai pada saatnya, karena kompiler menemukan kondisi-kondisi yang belum terpenuhi yang tidak bisa dikerjakan.

3 Kesalahan logika (*logical errors*), yaitu kesalahan dari logika program yang dibuat. Pengujian bentuk kesalahan ini dilakukan dengan *pengujian data*, yaitu dengan menjalankan program dengan data tertentu dan membandingkan hasil pengolahannya dengan hasil yang sudah diketahui. Proses melacak kesalahan ini dilakukan melalui *debugging*.

Pengujian program dilakukan dua tingkat pengujian, yaitu:

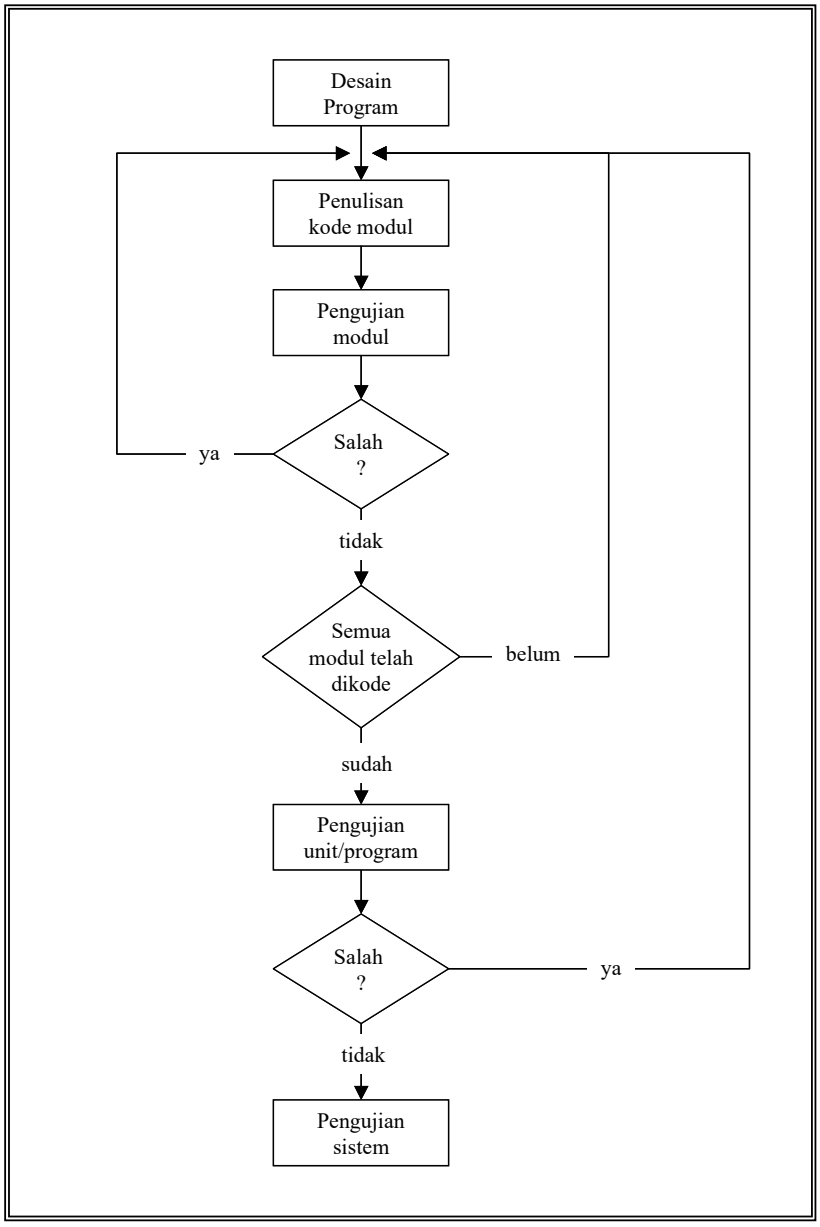
1 Pengujian modul

Pengujian untuk tiap modul program (berupa program utama, subroutine, dan subprogram) dikenal dengan istilah *stub testing*. Pengujian ini dilakukan dengan memberikan parameter-parameter yang diterima suatu modul, kemudian dilakukan pengujian atas hasil pengolahan untuk kemudian diuji kebenarannya.

2 Pengujian unit atau pengujian program

Pengujian program dilakukan setelah semua modul diuji secara independen sampai bebas dari kesalahan dan telah dirangkai menjadi satu unit program. Pengujian ini umumnya disebut sebagai unit testing atau program testing yang dimaksudkan untuk meyakinkan bahwa semua modul telah bekerja terintegrasi tanpa mengalami kesalahan.

Gambar V.23 menunjukkan diagram alir dalam melakukan langkah-langkah pengujian untuk setiap modul dan integrasi modul dalam suatu unit atau program.



Gambar V.23 Diagram Alir Pengujian Program

3.3 Pengujian Sistem

Validasi sistem merupakan kata kunci sukses bagi seorang pengambil keputusan untuk menggunakan suatu perangkat SPK (Borenstein, 1998). Pengujian sistem dilakukan setelah pengujian program. Pengujian sistem dilakukan untuk memeriksa kekompakan antar komponen sistem yang diimplementasikan. Tujuan utama dari pengujian sistem ini adalah untuk

memastikan bahwa elemen-elemen atau komponen-komponen dari sistem telah berfungsi sesuai dengan yang diharapkan. Pengujian perlu dilakukan untuk mencari kesalahan-kesalahan atau kelemahan-kelemahan yang mungkin terjadi. Kumpulan dari program-program yang telah diintegrasikan perlu diuji keandalannya, khususnya dalam hal input, pengolahan data, pengelolaan basis data, dan pengujian output sistem. Dalam penelitian ini, pengujian keandalan sistem dilakukan dengan studi kasus (hipotetik).

BAB VI
KESIMPULAN DAN SARAN

1 Kesimpulan

a SPKKM merupakan perangkat pengambilan keputusan yang menyediakan dukungan esensial dalam membantu pengambil keputusan dalam pembuatan struktur keputusan, analisis perferensi dari pengambil keputusan, dan penetapan alternatif. Perancangan SPKKM dalam penelitian ini ditujukan guna membantu pengambil keputusan dalam mempercepat dan mempermudah proses pengambilan keputusan idividu maupun kelompok, untuk memilih berbagai alternatif keputusan yang merupakan hasil pengolahan terhadap informasi-informasi yang diperoleh/tersedia dengan menggunakan model PKKM.

b Pendekatan pengembangan model untuk SPKKM merupakan suatu cara untuk mengembangkan hubungan-hubungan logis yang mendasari persoalan pengambilan keputusan ke dalam model matematis dan model informasi, yang mencerminkan hubungan yang terjadi diantara faktor-faktor yang terlibat.

c Model yang dikembangkan pada penelitian ini adalah pengembangan model AHP, dengan hasil berupa model keputusan bagi SPKKM. Model yang dikembangkan merupakan suatu kerangka metodologi penyelesaian masalah PKKM yang dalam analisisnya menggunakan prinsip dekomposisi, analisis perbandingan, analisis interdependensi kriteria, sintesa prioritas, analisis konsensus dan analisis pemilihan alternatif.

(1) Dekomposisi

Analisa masalah dimulai identifikasi objek keputusan, dan kemudian mengatur bagian-bagian dari komponen-komponen tersebut dalam bentuk hirarki tujuan.

(2) Analisis Perbandingan

Analisis perbandingan pada prinsipnya dilakukan dengan memberikan bobot pada alternatif. Evaluasi pembobotan alternatif dilakukan dengan menggunakan perbandingan preferensi antar elemen keputusan

(3) Analisis Interdependensi

Mengukur interdependensi kriteria, apabila antar kriteria memiliki hubungan interdependensi, maka perlu dilakukan penyesuaian nilai bobot terhadap masing-masing kriteria tersebut, sesuai dengan tingkat ketergantungannya

(4) Sintesa Prioritas

Sintesa prioritas dilakukan dengan membuat bobot pada masing-masing elemen hirarki tujuan. Pembobotan ini dilakukan atas dasar pertimbangan tingkat kepentingan kriteria yang berbeda untuk suatu persoalan, sintesa prioritas dilakukan dengan menghitung bobot (lokal dan global) kriteria dan alternatif.

(5) Analisis Konsensus

Metoda konsensus yang dikembangkan pada penelitian ini dapat memberikan informasi mengenai tingkat kesepakatan dan ketidaksepakatan, serta rentang ketidaksepakatan individu dan kelompok, struktur klaster, koherensi preferensi individu, dan koherensi elemen objek keputusan. Analisis ini memberikan informasi objek keputusan “bermasalah” dan pendapat individu yang marjinal. Pembentukkan konsesus dilakukan berdasarkan pendekatan metoda Delphi.

(6) Analisis Pemilihan Alternatif

Bagian akhir analisa keputusan dilakukan proses pemilihan alternatif dengan menggunakan programa linier.

d Gagasan dasar yang digunakan perancangan sistem pada SPKKM adalah hubungan antara data, model, dan keputusan yang dihasilkan. SPKKM dirancang berbasis seperangkat elemen (subsistem model, basis data, dialog, dan jaringan komunikasi) yang saling berinteraksi, membentuk suatu prosedur dalam pencapaian suatu tujuan atau tujuan-tujuan bersama, dengan mengoperasikan data untuk menghasilkan informasi, yang kemudian digunakan sebagai referensi dalam pengambilan keputusan.

2 Saran-Saran

a Perlu dilakukan penelitian lanjutan tentang analisis interpendensi kriteria dan alternatif.

b Perlu dikaji pendekatan pengambilan keputusan kelompok yang lebih baik. Hal ini didasari oleh pemikiran bahwa pendekatan pengambilan keputusan kolektif akan menghasilkan keputusan yang lebih baik dibandingkan dengan pemikiran individual. Untuk itu diperlukan instrumen yang dapat mengakomodasi kepentingan kelompok dengan lebih baik pada proses pengambilan sebuah keputusan.

c Perlu dilakukan penelitian selanjutnya untuk melihat lebih utuh tentang integrasi dan interaksi antar komponen penyusun SPK, yang terdiri dari subsistem model, basis data, dialog, dan jaringan komunikasi dalam membentuk sinergi elemen sistem.

DAFTAR PUSTAKA

Aktas, A. Z. (1987). Structured Analysis and Design of Information System. Prentice Hall, New York.

Al-Shemmeri, T., B. Al-Kloub, dan A. Pearman. (1997). Model Choice in Multicriteria Decision Aid. EJOR Vol. 97 No. 3, 16 March 1997. Elsevier Science Publisher B.V., North-Holland. p. 550-560.

Anonimus. (1997). Pemrograman Praktis dengan Delphi 2.0. Andi, Jogyakarta.

Anonimus. (2000). Modul Pelatihan; Analityc Hierarchy Process Berbantuan Perangkat Lunak "Expert Choice" for Windows. Teknik dan Manajemen Industi, Fakultas Teknik, Universitas Pasundan, Bandung.

Bazaraa, M. S., J.J. Jarvis, dan H. D. Shareli. (1990). Linear Programming and Network Flows. John Wiley and Sons, Inc., Canada.

Belton, V., dan J. Hodgkin. (1999). Facilitators, Decision Makers, D.I.Y. Users: Is Intelligent Multicriteria Decision Support for All Feasible or Desirable?. EJOR Vol. 113 No. 2, 1 March 1999. Elsevier Science Publisher B.V., North-Holland. p. 247-260.

Bodily, S.E. (1985). Modern Decision Making; A Guide to Modeling with Decision Support Systems. McGraw-Hill, Singapore.

Borenstein, D. (1998). Towards a Practical Method to Validate Decision Support System. DSS Vol. 23 No. 3, 1 July 1998. Elsevier Science Publisher B.V., North-Holland. p. 227-239.

Brans, J. P. (1994). The Space of Freedom of the Decision Maker or Modeling the Human Brain. Vrije Universiteit, Brussel.

Brans, J. P. dan B. Mareshal. (1992). Promethee V: MCDM Problems with Segmentation Constrain. INFOR Vol. 30 No. 2, May 1992. p. 85-96.

Brans, J. P. dan B. Mareshal. (1994). The Promcalc and GAIA Decision Support System for Multicriteria Decision Aid. EJOR, Elsevier Science Publisher B.V., Holland. p. 297-310.

Brans, J. P., Vinckle dan B. Mareshal. (1986). How to Select and How to Rank Projects; The Promethee Method. EJOR, Elsevier Science Publisher B.V., Holland. p. 228-238.

Briggs, T., P. L. Kunsch dan B. Mareschal. (1990). Nuclear Waste Management: An Application of the Multicriteria PROMETHEE Methods. EJOR, Elsevier Science Publisher B.V., Holland. p. 1 -10

Brodjonegoro, P.P.S. (1992). AHP. Pusat Antar Universitas-Studi Ekonomi, Universitas Indonesia, Jakarta.

Brown, R. V. (1974). Decision Analysis for the Manager. Holt, Rinehart and Winston, Inc.

Brugha, S. M. (1998). The Structure of Development Decision-Making. EJOR Vol. 104 No. 1, 1 January 1998. Elsevier Science Publisher B.V., North-Holland. p. 77-92.

Costa, C. A. B., L. Ensslin, É. C. Corréa, dan J. C. Vansnick. (1999). Decision Support Systems in Action: Integrated Application in a Multicriteria Decision Aid Process. EJOR Vol. 113 No. 2, 1 March 1999. Elsevier Science Publisher B.V., North-Holland. p. 315-335.

Davis, G. B. dan M. H. Oslo. (1984). Management Information Systems ; Conceptual, Foundations, Structure and Development. McGraw-Hill, Singapore.

DeCoster, D.T. dan E. L. Schafer. (1976). Financial Management; A Decision Emphasis. John Willey and Sons, Chichester.

Doumpos, M. dan C. Zopounidis. (2001). Assessing Financial Risks Using a Multicriteria Sorting Procedure: The Case of Country Risk Assessment. Omega Vol. 29 No. 1, 1 February 2001. 97-109

Driver, M.J. dan K. R. Brousseau. (1993). The Dinamic Decision Maker, Jossey Bass Publishers, San Fransisco.

Dunn, W. N. (1981). Public Policy Analysis; An Introduction. Prince-Hall, New Jersey.

Duta, A., dan S. Heda. (2000). Information Systems Architecture to Support Managed Care Businnes Process. DSS Vol. 30 No. 2, 27-December 2000**.** Elsevier Science Publisher B.V., North-Holland. p. 217-225

Easley, R. F., J. S. Valacich, dan M. A. Venkataramanan. (2000). Capturing Group Preferences in a Multicriteria Decision. EJOR Vol. 125 No. 1, 16 August 2000. Elsevier Science Publisher B.V., North-Holland. p. 73-83.

Edwards, C., J. Ward, dan A. Bytheway. (1991). The Essence of Information Systems. Prentice Hall International Ltd., New York.

Elerman, M.A., F. Neiderman, dan C. Adams. (1995). DSS Theory: A Model of Construct and Relationships. DSS Vol. 14 No. 1, 1 May 1995. North-Holland. p. 1-26.

Emery, F. E. (Ed). (1976). Systems Thinking. Penguin Books Ltd., Harmondsworrth.

Emshoff, J. R. dan R.L. Simon. (1970). Design and Use of Computer Simulation. Macmillan Publishing, New York.

Espinasse, B., G. Picolet, dan E. Chauraqui. (1997). Negoitation Support System and Multi Agent Approach. EJOR Vol. 103 No. 2, 1 December 1997. Elsevier Science Publisher B.V., North-Holland. p. 389-409.

Eum, Y. S., K. S. Park dan S. H. Kim. (2001). Establishing Dominance and Potential Optimality in Multi-Criteria Analysis with Imprecise Weight and Value. Computers and Operations Research, 1 April 2001. p. 397-409.

Feldt, A. G. Teori Perencanaan. di dalam. Catanese, A. J. dan J. C. Snyder (eds.). (1992). Perencanaan Kota. Erlangga, Jakarta.

Flood, R.L. dan E. R. Carson. (1990). Dealing with Complexity; An Intoduction to the Theory and Application of Systems Science. Plenum Press, New York.

Floyd, N. A. (1991). Essentials of Information Processing. Irwin, Boston.

Forman, E. H. dan Peniwati, K. (1998). Aggregating Individual Judgments and Priorities with the Analytic Hierarchy Process, EJOR Vol. 108 No. 1, 1 July 1998. Elsevier Science Publisher B.V., North-Holland. p. 165-169.

Forman, E. H., T. L. Saaty, M. A. Selly, R. Whitaker, J. Saaty dan F. Ruffing. (1986). Expert Choice Version 8; User Manual. Expert Choice, Inc., Pittsburgh.

Gandibleux, X. (1999). Interactive Multicriteria Procedure Exploiting a Knowledge-Based Module to Select Electricity Production Alternatives: The CASTART System. EJOR Vol. 113 No. 2, 1 March 1999. Elsevier Science Publisher B.V., North-Holland. p. 355-373.

Gibson, J.L., Ivancevich, J.M. dan Donelly, J.H. (1990). Organisasi dan Manajemen; PerilakuStruktur, dan Proses. Erlangga, Jakarta.

Gordon, G. (1989). System Simulation. Prince Hall of India Private Limited, New Delhi.

Guiltoni, A., dan J. M. Martel. (1998). Tentative Guidelines to Help Choosing an Appropiate MCDA Method. EJOR Vol. 109 No. 2, 1 September 1998. Elsevier Science Publisher B.V., North-Holland. p. 501-521.

Halim, M., U. B. Hidayat, dan Hidajat, J. (1996). Analisis Keputusan. Program Pascasarjana ITB, Bandung.

Harsey, P. and B. Ken. (1983). Management of Organizational Behavior. Prince Hall, New Delhi.

Hasibuan, M.S.P. (1987). Manajemen; Dasar, Pengertian, dan Masalah. Haji Masagung, Jakarta.

Horowitz, E. dan S. Sahni. (1978). Fundamentals of Computer Algoritms. Computer Science Press, Inc., Maryland.

Houben, G., K. Lenie., dan K. Vanhoof. (1999). A Knowledge-Based SWOT-Analysis System as an Instrument for Strategic Planning in Small and Medium Sized Enterprises. DSS Vol. 26 No. 2, 1 August 1999. Elsevier Science Publisher B.V., North-Holland. p. 125-135.

Jaszkiewicz, A., dan A. B. Ferhat. (1999). Solving Multiple Criteria Problems by Interactive Trichotomy Segmentation. EJOR Vol. 113 No. 2, 1 March 1999. Elsevier Science Publisher B.V., North-Holland. p. 271-280.

Jauch, L. R. dan W. F. Glueck. (1995). Manajemen Stategis dan Kebijakan Perusahaan. Erlangga, Jakarta.

Jogiyanto. (1990). Analisis dan Disain Sistem Informasi; Pendekatan Terstruktur. Andi, Jogyakarta.

Jones, C.V. (1995). Developments in Graph-Based Modelling for Decision Support. DSS Vol. 13 No. 1, 1 January 1995. North-Holland. p. 61-74.

Kanungo, S., S. Sharma, dan P. K. Jain. (2001). Evaluation of a Decision Support System for Credit Management Decisions. DSS Vol. 30 No. 4, 1 March 2001. Elsevier Science Publisher B.V., North-Holland. p. 419-436.

Karacapilidis, N. I., dan C. P. Pappis. (1997). A Framework for Group Decision Support Systems: Combining AI Tools and OR Techniques. EJOR Vol. 103 No. 2, 1 December 1997. Elsevier Science Publisher B.V., North-Holland. p. 373-388.

Kowal, J.A. (1991). Analysis Systems. Prentice Hall, Englewood Cliffs.

Kusumaningrum, A. D. (2000). Pengembangan Metode Pengukuran Kesepakatan untuk Pengambilan Keputusan dalam Kelompok. Thesis Magister. Institut Teknologi Bandung, Bandung.

Landry, M., J. L. Malouin, dan M. Oral. (1983). Model Validation in Operations Research. EJOR, Elsevier Science Publisher B.V., Holland. p. 204-220

Levin, R.I., D.S. Rubin. J. P. Stinson, dan E. S. Gardner. (1989). Pengambilan Keputusan Secara Kualitatif. Rajawali Press, Jakarta.

Li, D. (1990). A Conceptual Framework for Model Management System and its Applications in actDSS (III). Dissertation. Department of System Science, Tokyo Institute of Technology, Tokyo.

Liu, X. (1992). Systemic Structure-Oriented Approach to DSS Building. Dissertation. Department of System Science, Tokyo Institute of Technology, Tokyo.

Lucas, H. C. (1993). Analisis, Desain, dan Implementasi Sistem Informasi. Erlangga, Jakarta.

Ma, J., Z. P. Fan, dan L. H. Huang. (1999). A Subjective and Objective Integrated Approach to Determine Attribute Weights. EJOR Vol. 112 No. 2, 16 January 1999. Elsevier Science Publisher B.V., North-Holland. p. 379-404.

Malczewski, J., dan M. Jakson. (2000). Multicriteria Spatial Allocation of Educational Resources: An Overview. Sosio-Economic Planning Science Vol. 34 No. 3, 1 September 2000. p. 219-235.

Mangkusubroto, K. dan L. Tresnadi. (1987). Analisa Keputusan; Pendekatan Sistem dalam Manajemen Usaha dan Proyek. Ganeca Exact, Bandung.

Manly, H.J. (1957). Executive Decision Making. Richard D. Erwin, Inc., Illinois. didalam Hasibuan, M.S.P. (1987). Manajemen; Dasar, Pengertian, dan Masalah. Haji Masagung, Jakarta.

Mardiono, N.M.T. (1986). Penelitian Operasional; Teori dan Latihan. Dharma Patria, Bandung

Mareschal, B. dan J. P. Brans. (1991). BANKADVISER: An Industrial Evaluation System. EJOR, Elsevier Science Publisher B.V., Holland. p. 318-424.

Martin, J. (1992a). Information Engineering, Book I: Introduction. Prentice Hall, Englewood Cliffs.

Martin, J. (1992b). Information Engineering, Book II: Planing and Analysis. Prentice Hall, Englewood Cliffs.

Martin, J. (1992c). Information Engineering, Book III: Design and Construction. Prentice Hall, Englewood Cliffs.

Matcho, J. dan D. R. Faulkner. (1997). Edisi Khusus Panduan Penggunaan Delphi, Andi, Jogyakarta.

Matsatsinis, N. F., dan Y. Siskos. (1999). MARKEX: An intelligent Decision Support System for Product Development Decisions. EJOR Vol. 113 No. 2, 1 March 1999. Elsevier Science Publisher B.V., North-Holland. p. 336-354.

Murdick, R.G., J.E. Ross, dan J.R. Claggett. (1995). Sistem Informasi untuk Manajemen Modern. Erlangga, Jakarta.

Murhy, D.N.P., N.W. Page, dan E.Y. Rodin. (1990). Mathematical Modelling; A Tool for Problem Solving in Engineering, Physical, Biological and Social Sciences. Pergamon Press, Canada.

Murillo. M. A. V. (1990). A Production Management Decission Support System based on Theory of Hierachycal System. Dissertation. Department of System Science, Tokyo Institute of Technology, Tokyo.

Ngwenyama, O.K., N. Bryson, dan A. Mobolurin. (1996). Supporting Facilitations In Group Support System; Technique for Analyzing Consensus Relevant Data. DSS Journal. Vol 16 No. 2**,** 01 February 1996**.** p.155-168

O'leary, T. J. dan B. K. Williams. (1989). Computers and Information Systems. The Benjamin/Cummings Publishing Company Inc., Redwood City.

Orila, L. S. (1986). Computers and Information; An Introduction. McGraw-Hill Book Company, New York.

Pal, K., dan O. Palmer. (2000). A Decision-Support for Business Acquisitions. DSS Vol. 27 No. 4, 1 January 2000. Elsevier Science Publisher B.V., North-Holland. p. 411-429.

Parker, C. dan T. Case. (1993). Management Information Systems; Strategy and Action. McGraw-Hill, Singapore.

Phillips, D.T., A. Ravindran, dan J. Solberg. (1976). Operations Research; Principles and Practice. John Wiley and Sons, New York.

Power, D. J. (1999). A Brief History of Decision Support System. 22 March 1999. http://dssresources.com/history/dsshistory.html.

Power, D. J. (2000). Suporting Decision-Makers: An Expanded Framework. 15 December 2000. http://dssresources.com/papers/supportingdm/sld001.htm

Power, D.J. (1998). What is Decision Support System?. The On-Line Executive Journal for Data-Intensive Decision Support, 21 October 1998: Vol. 1, No. 3. http://dssresources.com/papers/whatisadss/index.html.

Pudinovski, V. V. (1999). A DSS for Multiple Criteria Decision Analysis with Imprecisely Specified Trade-Offs. EJOR Vol. 113 No. 2, 1 March 1999. Elsevier Science Publisher B.V., North-Holland. p. 261-270.

Putro, U. S., dan J. H. Tjakraatmadja. (1998). Proses Pengambilan Keputusan di dalam Situasi yang Interaktif dengan "Hypergame". Journal Teknik dan Manajemen Industri Vol. 18 No. 2. Agustus 1998. Departemen Teknik Industri, Institut Teknologi Bandung, Bandung. p. 48-59.

Raghunathan, S. (1996). A Stuctured Modelling Based Methodology to Design Decision Support System. DSS Vol. 17 No. 4, 2 July 1996. North-Holland. p. 299-312.

Rajabi, S., D. M. Kilgour, dan K. W. Hipel. (1998). Modelling Action Interdependence in Multiple Criteria Decision Making. EJOR Vol. 110 No. 3, 1 Nopember 1998. Elsevier Science Publisher B.V., North-Holland. p. 490-508

Ramdhani, M.A. (1997). Penetapan Prioritas Lokasi Perumahan Berdasarkan Penggabungan Metoda PROMETHEE dan AHP. Thesis. Program Magister Teknik Manajemen Industri, ITB, Bandung.

Robbins, S. P. (1993). Organizational Behavior; Concept, Conroversies, and Aplication. Price Hall International.

Robert, L.F. dan C.J. Michael. (1991). Creative Problem Solving. John Willey and Sons, Canada.

Robichaud, B., E. J. Muscat, dan A. M. Hall. (1989). Introduction to Information Processing. McGraw-Hill Book Company, New York.

Saaty, T. L. (1994). Fundamentals of Decision Making and Priority Theory with the Analytical Hierarchy Process. The Analytical Hierarchy Process Series Vol. VI. RWS Publications, Pittsburgh.

Saaty, T. L. (1994). Decision Making in Economic, Political, Social and Technological Environment with the Analytical Hierarchy Process. The Analytical Hierarchy Process Series Vol. VII. RWS Publications, Pittsburgh.

Saaty, T.L. (1996). Decision Making with Dependence and Feedback; The Analytic Network Process. RWS Publications, Pittsburgh.

Sage, A. P. (1990). Decision Support System. di dalam. Sage, A.P. (ed.). (1990. Concise Encyclopedia of Information Processing in Systems and Organization. Pergamon, Manchester.

Sage. A. P. (ed). (1989). Concise Encyclopedia of Information Processing in Systems and Organizations. Pergamon Press, Manchester.

Salusu, J. (1996). Pengambilan Keputusan Strategik untuk Organisasi Publik dan Organisasi Non Profit. Gramedia, Jakarta.

Sarkis, J. (2000). A Comparative Analysis of DEA as a Discrete Alternative Multiple Criteria Decision Tool. EJOR Vol. 123 No. 3, 16 June 2000. Elsevier Science Publisher B.V., North-Holland. p. 543-557.

Sawicki, D. S. Analisis Kebijakan. didalam. Catanese, A. J. dan J. C. Snyder (eds.). (1992). Perencanaan Kota. Erlangga, Jakarta.

Shaw, J.E., S. J. Mecca, dan M. Redick. (1996). Problem Solving; System Approach. Petrocee Book.

Simatupang, T. M. (1994). Pemodelan Sistem. Nindita, Klaten.

Simon, H. (1960). The New Science of Management Decision. Harper and Row, New York.

Siregar, A. B. (1991). Pemodelan Sistem. Journal Teknik dan Manajemen Industri No. 5, April 1991. Departemen Teknik Industri, Institut Teknologi Bandung, Bandung.

Siskos, Y. (1999). Managing Multiple Criteria via Intelligent Decision Support Systems. EJOR Vol. 113 No. 2, 1 March 1999. Elsevier Science Publisher B.V., North-Holland. p. 235.

Siskos, Y., A. Spyridakos, dan D. Yannacopoulos. (1999). Using Artificial Intelligence and Visual Technique into Preference Disaggregation Analysis: The MIIDAS System. EJOR Vol. 113 No. 2, 1 March 1999. Elsevier Science Publisher B.V., North-Holland. p. 281-299.

Siskos, Y., dan A. Spyridikos. (1999). Intelligent Multicriteria Decision Support: Overview and Perspectives. EJOR Vol. 113 No. 2, 1 March 1999. Elsevier Science Publisher B.V., North-Holland. p. 236-246.

Slotnick, D. L., E. M. Butterfield, E. S. Colantonio, D. J. Kopetzky, dan J. K. Slotnick. (1986). Computers and Applications; An Introduction to Data Processing. D.C. Heath and Company, Lexington.

Sprague, R. H., dan E. D. Carlson (1982). Building Effective Decision Support Systems. Prentice-Hall, New Jersey.

Stewart, T. J. (1999). Evaluation and Refinement of Aspiration-Based Methods in MCDM. EJOR Vol. 113 No. 3, 16 March 1999. Elsevier Science Publisher B.V., North-Holland. p. 643-652.

Stoner, J.A.F. dan C. Wankel. (1993). Perencanaan dan Pengambilan Keputusan dalam Manajemen. Rineka Cipta, Jakarta.

Sudirman, I. (1998). Pengaruh Penggunaan Informasi dan Fokus pada Gaya Pengambilan Keputusan. Jurnal TMI Volume 18 No. 1, April 1998. Program Studi Teknik dan Manajemen Industri, Program Pascasarjana, Intitut Teknologi Bandung, Bandung. p. 7-13.

Sudirman, I. dan Widjajani. (1996). Sistem Informasi Manajemen. LEMLIT UNPAS Press, Bandung.

Suryadi, K. (1992). Vers Une Integration des Fonctions de Planification et D'ordonnancement pour L'aide a la Decision en Gestion de Production. Ph.D. Thesis. Universite d'Aix Marseille-3, France.

Suryadi, K. dan E. R. Nuzal. (1998). A Decision Support System for Car Fault Diagnosis Using Expert System. Information Sciences for Decision Making, Paris. No. 2 April 1998. p. 75-78

Suryadi, K., dan M. A. Ramdhani. (1998). Sistem Pendukung Keputusan; Suatu Wacana Struktural Idealisasi dan Implementasi Konsep Pengambilan Keputusan. Remaja Rosda Karya, Bandung.

Syamsi, I. (1995). Pengambilan Keputusan dan Sistem Informasi. Bumi Aksara, Jakarta.

Taha, H. A. (1992). Operations Research; An Introduction. Macmillan Publishing Company, New York.

Thierauf, R.J. (1982). Decision Support Systems for Effective Planning and Control. Prince Hall, New Jersey.

Vickle, P., M. Gassner, dan B. Roy. (1992). Multicriteria Decision Aid. John Willey and Sons, Chichester.

Volberda, H. W. dan A. Rutges. (1999). FARSYS: A Knowledge-Based System for Managing Strategic Change. DSS Vol. 26 No. 2, 1 August 1999. Elsevier Science Publisher B.V., North-Holland. p. 99-123.

Welch, S. dan J. C. Comer. (1983). Quantitative Methods for Public Administration, Technique and Applications. The Dorsey Press, Illinois.

Wierzbicki, A. P., dan J. Granat. (1999). Multi-Objective Modelling for Engineering Applications: DIDASN++ system. EJOR Vol. 113 No. 2, 1 March 1999. Elsevier Science Publisher B.V., North-Holland. p. 374-389.

Zahir, S. (1999). Clusters in a Group: Decision Making in the Vector Space Formulation of the Analytic Hierarchy Process. EJOR Vol. 112 No. 3, 01 February 1999. Elsevier Science Publisher B.V., North-Holland. p. 620-634

Zahir, S. (1999). Geometry of Decision Making and the Vektor Space Formulation of the Analytic Hierarchy Process. EJOR Vol. 112 No. 2, 01 January 1999. Elsevier Science Publisher B.V., North-Holland. p. 374-389

Zeleny, M. (1982). Multiple Criteria Decision Making. McGraw-Hill, New York.

Zompounidis, C. dan M. Doumpos. (2000). PREFDIS: A Multicriteria Decision Support System for Sorting Decision Problems. Computer and Operation Research Journal. Vol. 27. Elsevier Science Publisher B.V., North-Holland. p. 779-797.

RIWAYAT HIDUP

Muhammad Ali Ramdhani dilahirkan di Garut, Jawa Barat pada tanggal 6 Nopember 1971 sebagai putra ketiga dari enam bersaudara, dari orang tua Prof. H. Cecep Syarifuddin dan Dr. Hj. Ummu Salamah, M.S., beristrikan Hj. Hilda Ainis Syifa dan memiliki satu putra Muhammad Khalifa Umana.

Memasuki dunia pendidikan formal dimulai pada jenjang Sekolah Dasar Negeri Kiansantang, Garut lulus pada tahun 1984, Sekolah Menengah Tingkat Pertama Negeri 2 Garut lulus pada tahun 1987, Sekolah Menengah Tingkat Atas Negeri 12 Bandung lulus tahun 1990. Kemudian dilanjutkan ke jenjang perguruan tinggi di Institut Pertanian Bogor pada Jurusan Teknologi Industri Pertanian, lulus pada tahun 1994 dengan gelar kesarjanaan Sarjana Teknologi Pertanian (S.TP.).

Pada tahun 1994, menjadi Asisten Praktikum untuk mata ajaran Menggambar Teknik dan Teknologi Hasil Pengolahan Hewan dan Perairan, pada Jurusan Teknologi Industri Pertanian, IPB. Sejak Desember tahun 1994 diangkat sebagai pegawai tetap pada Yayasan Al-Musaddadiyah yang dipekerjakan sebagai tenaga edukatif pada Jurusan Teknik dan Manajemen Industri, Sekolah Tinggi Teknologi Garut. Penulis menjadi Dosen Tamu di Program Magister Teknik dan Manajemen Industri, Institut Teknologi Bandung (1998-2000), Staf Pengajar Luar Biasa pada Sekolah Tinggi Teknologi Bandung (1996), Fakultas Ekonomi dan Fakultas Ilmu Sosial dan Ilmu Politik Universitas Garut (1995-sekarang).

Pada tahun 1995 mengikuti studi lanjutan S2 pada Program Pascasarjana, Institut Teknologi Bandung, Program Studi Teknik dan Manajemen Industri dengan bidang kajian utama Manajemen Industri, lulus pada tahun 1997 dengan gelar kesarjanaan Magister Teknik (M.T.). Pada tahun 1997 melanjutkan studi lanjutan S3 pada Program Pascasarjana, Institut Teknologi Bandung, Program Studi Teknik dan Manajemen Industri.

LAMPIRAN-LAMPIRAN

PERANCANGAN
SISTEM PENDUKUNG KEPUTUSAN KRITERIA MAJEMUK
PADA PENGAMBILAN KEPUTUSAN KELOMPOK

DISERTASI

Karya Tulis sebagai salah satu syarat
untuk memperoleh gelar Doktor dari
Institut Teknologi Bandung

Oleh
MUHAMMAD ALI RAMDHANI
NIM 322 97 009

INSTITUT TEKNOLOGI BANDUNG
2001

LAMPIRAN

A Rational Delphi Project File

B Rational Delphi Program File

Lampiran A Rational Delphi Project File (*.dpr)

```
program Rational3;

uses
  Forms,
  FMain in 'FMain.pas' {Form1},
  UDataStru in 'UDataStru.pas',
  USubject in 'USubject.pas',
  UMsgList in 'UMsgList.pas',
  KFieldObjects in 'KFieldObjects.pas',
  UDefault in 'UDefault.pas',
  UTreeDisplay in 'UTreeDisplay.pas',
  Fdlgdat in 'fdlgdat.PAS' {DialogData},
  Fpilihan in 'FPilihan.PAS' {FormPilihan},
  Fquest2 in 'FQuest2.PAS' {FormQuest2},
  GlobalUnit in 'GlobalUnit.pas',
  Fvalid in 'FValid.PAS' {FormValidasi},
  Fguide in 'FGuide.PAS' {FormGuide},
  Findep in 'Findep.pas' {FormInterDependency},
  Fpanduan in 'Fpanduan.pas' {FormPanduan},
  SPNL in 'SPnl.pas',
  Fsens in 'fSens.PAS' {FormSensitivitas},
  Fgsens in 'Fgsens.pas' {FormSensGraph},
  Tokutils in 'Tokutils.pas',
  Fprogln in 'Fprogln.pas' {FormProgramaLinier},
  Splash in 'Splash.PAS' {FormSplash},
  Fdocinfo in 'Fdocinfo.pas' {FormDocInfo},
  Gomory in 'Gomory.pas',
  FPrintHirarki in 'FPrintHirarki.pas' {FormPrintHirarki};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TDialogData, DialogData);
  Application.CreateForm(TFormPilihan, FormPilihan);
  Application.CreateForm(TFormQuest2, FormQuest2);
  Application.CreateForm(TFormValidasi, FormValidasi);
  Application.CreateForm(TFormGuide, FormGuide);
  Application.CreateForm(TFormInterDependency, FormInterDependency);
  Application.CreateForm(TFormPanduan, FormPanduan);
  Application.CreateForm(TFormSensitivitas, FormSensitivitas);
  Application.CreateForm(TFormSensGraph, FormSensGraph);
  Application.CreateForm(TFormProgramaLinier, FormProgramaLinier);
  Application.CreateForm(TFormSplash, FormSplash);
  Application.CreateForm(TFormDocInfo, FormDocInfo);
  Application.CreateForm(TFormPrintHirarki, FormPrintHirarki);
  Application.Run;
end.
```

```
program RatioIdv;

uses
  Forms,
  FMain in 'FMain.pas' {Form1},
  UDataStru in 'UDataStru.pas',
  USubject in 'USubject.pas',
  UMsgList in 'UMsgList.pas',
  KFieldObjects in 'KFieldObjects.pas',
  UDefault in 'UDefault.pas',
  UTreeDisplay in 'UTreeDisplay.pas',
  Fdlgdat in 'fdlgdat.PAS' {DialogData},
  Fpilihan in 'FPilihan.PAS' {FormPilihan},
  Fquest2 in 'FQuest2.PAS' {FormQuest2},
  GlobalUnit in 'GlobalUnit.pas',
  Fvalid in 'FValid.PAS' {FormValidasi},
  Fguide in 'FGuide.PAS' {FormGuide},
  Findep in 'Findep.pas' {FormInterDependency},
  Fpanduan in 'Fpanduan.pas' {FormPanduan},
  SPNL in 'SPnl.pas',
  Fsens in 'fSens.PAS' {FormSensitivitas},
  Fgsens in 'Fgsens.pas' {FormSensGraph},
  Tokutils in 'Tokutils.pas',
  Fprogln in 'Fprogln.pas' {FormProgramaLinier},
  Splash in 'Splash.PAS' {FormSplash},
  Fdocinfo in 'Fdocinfo.pas' {FormDocInfo},
  Gomory in 'Gomory.pas',
  FPrintHirarki in 'FPrintHirarki.pas' {FormPrintHirarki};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TDialogData, DialogData);
  Application.CreateForm(TFormPilihan, FormPilihan);
  Application.CreateForm(TFormQuest2, FormQuest2);
  Application.CreateForm(TFormValidasi, FormValidasi);
  Application.CreateForm(TFormGuide, FormGuide);
  Application.CreateForm(TFormInterDependency, FormInterDependency);
  Application.CreateForm(TFormPanduan, FormPanduan);
  Application.CreateForm(TFormSensitivitas, FormSensitivitas);
  Application.CreateForm(TFormSensGraph, FormSensGraph);
  Application.CreateForm(TFormProgramaLinier, FormProgramaLinier);
  Application.CreateForm(TFormSplash, FormSplash);
  Application.CreateForm(TFormDocInfo, FormDocInfo);
  Application.CreateForm(TFormPrintHirarki, FormPrintHirarki);
  Form1.SetupIndividual;
  Application.Run;
end.
```

```
program RatioHie;

uses
  Forms,
  FMain in 'FMain.pas' {Form1},
  UDataStru in 'UDataStru.pas',
  USubject in 'USubject.pas',
  UMsgList in 'UMsgList.pas',
  KFieldObjects in 'KFieldObjects.pas',
  UDefault in 'UDefault.pas',
  UTreeDisplay in 'UTreeDisplay.pas',
  Fdlgdat in 'fdlgdat.PAS' {DialogData},
  Fpilihan in 'FPilihan.PAS' {FormPilihan},
  Fquest2 in 'FQuest2.PAS' {FormQuest2},
  GlobalUnit in 'GlobalUnit.pas',
  Fvalid in 'FValid.PAS' {FormValidasi},
  Fguide in 'FGuide.PAS' {FormGuide},
  Findep in 'Findep.pas' {FormInterDependency},
  Fpanduan in 'Fpanduan.pas' {FormPanduan},
  SPNL in 'SPnl.pas',
  Fsens in 'fSens.PAS' {FormSensitivitas},
  Fgsens in 'Fgsens.pas' {FormSensGraph},
  Tokutils in 'Tokutils.pas',
  Fprogln in 'Fprogln.pas' {FormProgramaLinier},
  Splash in 'Splash.PAS' {FormSplash},
  Fdocinfo in 'Fdocinfo.pas' {FormDocInfo},
  Gomory in 'Gomory.pas',
  FPrintHirarki in 'FPrintHirarki.pas' {FormPrintHirarki};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TDialogData, DialogData);
  Application.CreateForm(TFormPilihan, FormPilihan);
  Application.CreateForm(TFormQuest2, FormQuest2);
  Application.CreateForm(TFormValidasi, FormValidasi);
  Application.CreateForm(TFormGuide, FormGuide);
  Application.CreateForm(TFormInterDependency, FormInterDependency);
  Application.CreateForm(TFormPanduan, FormPanduan);
  Application.CreateForm(TFormSensitivitas, FormSensitivitas);
  Application.CreateForm(TFormSensGraph, FormSensGraph);
  Application.CreateForm(TFormProgramaLinier, FormProgramaLinier);
  Application.CreateForm(TFormSplash, FormSplash);
  Application.CreateForm(TFormDocInfo, FormDocInfo);
  Application.CreateForm(TFormPrintHirarki, FormPrintHirarki);
  Form1.SetupHierarchy;
  Application.Run;
end.
```

```
program RatioQST;

uses
  Forms,
  FMain in 'FMain.pas' {Form1},
  UDataStru in 'UDataStru.pas',
  USubject in 'USubject.pas',
  UMsgList in 'UMsgList.pas',
  KFieldObjects in 'KFieldObjects.pas',
  UDefault in 'UDefault.pas',
  UTreeDisplay in 'UTreeDisplay.pas',
  Fdlgdat in 'fdlgdat.PAS' {DialogData},
  Fpilihan in 'FPilihan.PAS' {FormPilihan},
  Fquest2 in 'FQuest2.PAS' {FormQuest2},
  GlobalUnit in 'GlobalUnit.pas',
  Fvalid in 'FValid.PAS' {FormValidasi},
  Fguide in 'FGuide.PAS' {FormGuide},
  Findep in 'Findep.pas' {FormInterDependency},
  Fpanduan in 'Fpanduan.pas' {FormPanduan},
  SPNL in 'SPnl.pas',
  Fsens in 'fSens.PAS' {FormSensitivitas},
  Fgsens in 'Fgsens.pas' {FormSensGraph},
  Tokutils in 'Tokutils.pas',
  Fprogln in 'Fprogln.pas' {FormProgramaLinier},
  Splash in 'Splash.PAS' {FormSplash},
  Fdocinfo in 'Fdocinfo.pas' {FormDocInfo},
  Gomory in 'Gomory.pas',
  FPrintHirarki in 'FPrintHirarki.pas' {FormPrintHirarki};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TDialogData, DialogData);
  Application.CreateForm(TFormPilihan, FormPilihan);
  Application.CreateForm(TFormQuest2, FormQuest2);
  Application.CreateForm(TFormValidasi, FormValidasi);
  Application.CreateForm(TFormGuide, FormGuide);
  Application.CreateForm(TFormInterDependency, FormInterDependency);
  Application.CreateForm(TFormPanduan, FormPanduan);
  Application.CreateForm(TFormSensitivitas, FormSensitivitas);
  Application.CreateForm(TFormSensGraph, FormSensGraph);
  Application.CreateForm(TFormProgramaLinier, FormProgramaLinier);
  Application.CreateForm(TFormSplash, FormSplash);
  Application.CreateForm(TFormDocInfo, FormDocInfo);
  Application.CreateForm(TFormPrintHirarki, FormPrintHirarki);
  Form1.SetupQuestionnaire;
  Application.Run;
end.
```

```
program RatioGRP;

uses
  Forms,
  FMain in 'FMain.pas' {Form1},
  UDataStru in 'UDataStru.pas',
  USubject in 'USubject.pas',
  UMsgList in 'UMsgList.pas',
  KFieldObjects in 'KFieldObjects.pas',
  UDefault in 'UDefault.pas',
  UTreeDisplay in 'UTreeDisplay.pas',
  Fdlgdat in 'fdlgdat.PAS' {DialogData},
  Fpilihan in 'FPilihan.PAS' {FormPilihan},
  Fquest2 in 'FQuest2.PAS' {FormQuest2},
  GlobalUnit in 'GlobalUnit.pas',
  Fvalid in 'FValid.PAS' {FormValidasi},
  Fguide in 'FGuide.PAS' {FormGuide},
  Findep in 'Findep.pas' {FormInterDependency},
  Fpanduan in 'Fpanduan.pas' {FormPanduan},
  SPNL in 'SPnl.pas',
  Fsens in 'fSens.PAS' {FormSensitivitas},
  Fgsens in 'Fgsens.pas' {FormSensGraph},
  Tokutils in 'Tokutils.pas',
  Fprogln in 'Fprogln.pas' {FormProgramaLinier},
  Splash in 'Splash.PAS' {FormSplash},
  Fdocinfo in 'Fdocinfo.pas' {FormDocInfo},
  Gomory in 'Gomory.pas',
  FPrintHirarki in 'FPrintHirarki.pas' {FormPrintHirarki};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TDialogData, DialogData);
  Application.CreateForm(TFormPilihan, FormPilihan);
  Application.CreateForm(TFormQuest2, FormQuest2);
  Application.CreateForm(TFormValidasi, FormValidasi);
  Application.CreateForm(TFormGuide, FormGuide);
  Application.CreateForm(TFormInterDependency, FormInterDependency);
  Application.CreateForm(TFormPanduan, FormPanduan);
  Application.CreateForm(TFormSensitivitas, FormSensitivitas);
  Application.CreateForm(TFormSensGraph, FormSensGraph);
  Application.CreateForm(TFormProgramaLinier, FormProgramaLinier);
  Application.CreateForm(TFormSplash, FormSplash);
  Application.CreateForm(TFormDocInfo, FormDocInfo);
  Application.CreateForm(TFormPrintHirarki, FormPrintHirarki);
  Form1.SetupGroup;
  Application.Run;
end.
```

Lampiran B Rational Delphi Program File (*.pas)

```
unit FMain;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  ExtCtrls, ToolWin, ComCtrls, UDataStru, KFieldObjects, StdCtrls,
Math,
  UTreeDisplay, ActnList, ImgList, Menus;

type
  TForm1 = class(TForm)
    ToolBar1: TToolBar;
    Panel1: TPanel;
    Panel2: TPanel;
    ScrollBox1: TScrollBox;
    PaintBox1: TPaintBox;
    Memo1: TMemo;
    ActionList1: TActionList;
    ActAddCriteria: TAction;
    ActDeleteCriteria: TAction;
    ActModifyCriteria: TAction;
    ActAddAlt: TAction;
    ActDeleteAlt: TAction;
    ActModifyAlt: TAction;
    ToolButton1: TToolButton;
    ToolButton2: TToolButton;
    ToolButton3: TToolButton;
    ToolButton4: TToolButton;
    ToolButton5: TToolButton;
    ToolButton6: TToolButton;
    ToolButton7: TToolButton;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    ActPreference: TAction;
    ToolButton8: TToolButton;
    ToolButton9: TToolButton;
    Button4: TButton;
    ToolButton10: TToolButton;
    ActIndep: TAction;
    ToolButton11: TToolButton;
    ToolButton12: TToolButton;
    ToolButton13: TToolButton;
    ToolButton14: TToolButton;
    ToolButton15: TToolButton;
    ImageList1: TImageList;
    ActNewFile: TAction;
    ActLoadFile: TAction;
    ActSaveAs: TAction;
    ActSave: TAction;
    OpenDialog1: TOpenDialog;
```

```
SaveDialog1: TSaveDialog;
ToolButton16: TToolButton;
ActSens: TAction;
ToolButton17: TToolButton;
ActGSens: TAction;
ActDocInfo: TAction;
MainMenu1: TMainMenu;
File1: TMenuItem;
Exit1: TMenuItem;
N1: TMenuItem;
PrintSetup1: TMenuItem;
N2: TMenuItem;
SaveAs1: TMenuItem;
Save1: TMenuItem;
Open1: TMenuItem;
New1: TMenuItem;
DocumentInfo1: TMenuItem;
N3: TMenuItem;
Edit1: TMenuItem;
InsertCriteria1: TMenuItem;
EditCriteria1: TMenuItem;
DeleteCriteria1: TMenuItem;
N4: TMenuItem;
InsertaAlternative1: TMenuItem;
EditAlternative1: TMenuItem;
DeleteAlternative1: TMenuItem;
Analysis1: TMenuItem;
Compare1: TMenuItem;
Interdependency1: TMenuItem;
SensitivityTrace1: TMenuItem;
SensitivityGraph1: TMenuItem;
ActProgLin: TAction;
ToolButton18: TToolButton;
ToolButton19: TToolButton;
ToolButton20: TToolButton;
ToolButton21: TToolButton;
ToolButton22: TToolButton;
ToolButton23: TToolButton;
ToolButton24: TToolButton;
ToolButton25: TToolButton;
ToolButton26: TToolButton;
ToolButton27: TToolButton;
ActPrintSetup: TAction;
ActCetakHier: TAction;
ActCetakHierList: TAction;
ActCetakDesc: TAction;
ActJudgement: TAction;
ActReportWeight: TAction;
ActReportIndep: TAction;
ActReportSens: TAction;
ActReportProglin: TAction;
procedure FormCreate(Sender: TObject);
procedure FormDestroy(Sender: TObject);
procedure SetToRoot(Sender: TObject);
procedure PaintBox1Paint(Sender: TObject);
```

```
    procedure PaintBox1MouseDown(Sender: TObject; Button:
TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure ActAddCriteriaExecute(Sender: TObject);
    procedure ActDeleteCriteriaExecute(Sender: TObject);
    procedure ActModifyCriteriaExecute(Sender: TObject);
    procedure ActAddAltExecute(Sender: TObject);
    procedure ActDeleteAltExecute(Sender: TObject);
    procedure ActModifyAltExecute(Sender: TObject);
    procedure SetNextCrit(Sender: TObject);
    procedure SetPrevCrit(Sender: TObject);
    procedure ActPreferenceExecute(Sender: TObject);
    procedure PrepareProcess;
    procedure ActIndepExecute(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure ActNewFileExecute(Sender: TObject);
    procedure ActLoadFileExecute(Sender: TObject);
    procedure ActSaveExecute(Sender: TObject);
    procedure ActSaveAsExecute(Sender: TObject);
    procedure ActSensExecute(Sender: TObject);
    procedure RefreshDataClick(Sender: TObject);
    procedure ActGSensExecute(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure ActProgLinExecute(Sender: TObject);
    procedure ActDocInfoExecute(Sender: TObject);
    procedure ActPrintSetupExecute(Sender: TObject);
    procedure ActCetakHierExecute(Sender: TObject);
    procedure ActCetakHierListExecute(Sender: TObject);
  private
    { Private declarations }
    FDataStru: TRDataStru;
    FTreeDisplayer: TRTreeDisplayer;
    FXLeft: Integer;
    FMaxWidth: Integer;
    FArray: Array[0..10] of TPoint;
    FWindowType: Integer;
    FAllowLeaf: Boolean;
    FileName: string;
    procedure CalculateData;
    procedure A0NodeProcess(vCrit: TKFieldObject; vLevel: Integer);
    procedure A0PreProcess(vCrit: TKBlockObject; vLevel: Integer);
    procedure A0PostProcess(vCrit: TKBlockObject; vLevel: Integer);
  public
    { Public declarations }
    procedure SetupIndividual;
    procedure SetupHierarchy;
    procedure SetupGroup;
    procedure SetupQuestionnaire;
  end;

var
  Form1: TForm1;
```

```
implementation

uses Fdlgdat, Fpilihan, Fquest2, Findep, Fsens, FGSens, Fprogln,
Fdocinfo,
  FPrintHirarki;

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
var x0, x1, x2: TKBlockObject;
begin
  Panel1.Caption := '';
  OpenDialog1.InitialDir := ExtractFilePath(Application.ExeName);
  SaveDialog1.InitialDir := ExtractFilePath(Application.ExeName);
  FDataStru := TRDataStru.Create;
(*  x0 := FDataStru.AddCriteria(nil, 'Test1', '');
  x2 := x0;
  x1 := FDataStru.AddCriteria(x0, 'Test11', '');
  FDataStru.AddCriteria(x1, 'Test111', '');
  FDataStru.AddCriteria(x1, 'Test112', '');
  FDataStru.AddCriteria(x1, 'Test113', '');
  FDataStru.AddCriteria(x1, 'Test114', '');
  x1 := FDataStru.AddCriteria(x0, 'Test12', '');
  {FDataStru.AddCriteria(x1, 'Test121', '');
  FDataStru.AddCriteria(x1, 'Test122', '');
  FDataStru.AddCriteria(x1, 'Test123', '');}

  x0 := FDataStru.AddCriteria(nil, 'Test2', '');
  x1 := FDataStru.AddCriteria(x0, 'Test21', '');
  FDataStru.AddCriteria(x1, 'Test211', '');
  FDataStru.AddCriteria(x1, 'Test212', '');
  FDataStru.AddCriteria(x1, 'Test213', '');
  x1 := FDataStru.AddCriteria(x0, 'Test22', '');
  FDataStru.AddCriteria(x1, 'Test221', '');
  FDataStru.AddCriteria(x1, 'Test222', '');
  FDataStru.AddCriteria(x1, 'Test223', '');

  x0 := FDataStru.AddCriteria(nil, 'Test3', '');
  x1 := FDataStru.AddCriteria(x0, 'Test31', '');
  FDataStru.AddCriteria(x1, 'Test311', '');
  FDataStru.AddCriteria(x1, 'Test312', '');
  FDataStru.AddCriteria(x1, 'Test313', '');
  x1 := FDataStru.AddCriteria(x0, 'Test32', '');
  FDataStru.AddCriteria(x1, 'Test321', '');
  FDataStru.AddCriteria(x1, 'Test322', '');
  FDataStru.AddCriteria(x1, 'Test323', '');
  FDataStru.AddAlternative('Alt1', '');
  FDataStru.AddAlternative('Alt2', '');
  FDataStru.AddAlternative('Alt3', '');
  FDataStru.AddAlternative('Alt4', '');
  FDataStru.AddAlternative('Alt5', '');    *)
```

```
//FDataStru.SaveToFile(ExtractFilePath(Application.ExeName)+'test.xx
x');
  FTreeDisplayer := TRTreeDisplayer.Create;
  FTreeDisplayer.Canvas := PaintBox1.Canvas;
  FTreeDisplayer.DataStru := FDataStru;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  FDataStru.Free;
end;

procedure TForm1.SetToRoot(Sender: TObject);
begin
  FTreeDisplayer.SelCriteria := FDataStru.Root;
  PaintBox1.Refresh;
end;

procedure TForm1.A0NodeProcess(vCrit: TKFieldObject; vLevel:
Integer);
begin
  //PaintBox1.Canvas.Rectangle(vCrit.TagValue1
end;

procedure TForm1.A0PreProcess(vCrit: TKBlockObject; vLevel:
Integer);
begin
  FArray[vLevel].x := 99999;
  FArray[vLevel].y := 0;
{  if vCrit.OwnerField.Owner.OwnerField<>nil then begin
    Memo1.Lines.Add('2-
'+vCrit.OwnerField.Owner.OwnerField.FieldName+
    '-'+IntToStr(vCrit.FieldCount));
  end else begin
    Memo1.Lines.Add('2-Goal');
  end;}
end;

procedure TForm1.A0PostProcess(vCrit: TKBlockObject; vLevel:
Integer);
var vx, vy: Integer;
  vFIeld: TKFieldObject;
  i: Integer;
begin
  if vCrit.OwnerField.Owner.OwnerField<>nil then begin
    vFIeld := vCrit.OwnerField.Owner.OwnerField;
    {Memo1.Lines.Add('3-'+vField.FieldName+
      '-'+IntToStr(vCrit.FieldCount));}
  end else begin
    //Memo1.Lines.Add('3-Goal');
    vField := nil;
  end;
  if (vField<>nil) then begin
    if (vCrit.FieldCount=0) then begin
      vx := FXLeft;
      vy := vLevel*80;
```

```
      PaintBox1.Canvas.Rectangle(vx, vy, vx+80, vy+50);
      PaintBOx1.Canvas.TextOut(vx, vy, vField.FieldName);
      FXLeft := FXLeft+100;
      if vLevel>0 then begin
        FArray[vLevel-1].x := Min(FArray[vLevel-1].x, vx);
        FArray[vLevel-1].y := Max(FArray[vLevel-1].y, vx);
      end;
      //FMaxWidth := Max(FMaxWidth, );
    end else begin
      vx := (FArray[vLevel].x+FArray[vLevel].y) div 2;
      vy := vLevel*80;
      PaintBox1.Canvas.Rectangle(vx, vy, vx+80, vy+50);
      PaintBOx1.Canvas.TextOut(vx, vy, vField.FieldName);
      if vLevel>0 then begin
        FArray[vLevel-1].x := Min(FArray[vLevel-1].x, vx);
        FArray[vLevel-1].y := Max(FArray[vLevel-1].y, vx);
      end;
    end;
  end else begin
    vx := (FArray[0].x+FArray[0].y) div 2;
    vy := vLevel*80;
    PaintBox1.Canvas.Rectangle(vx, vy, vx+80, vy+50);
    PaintBOx1.Canvas.TextOut(vx, vy, 'Goal');
  end;
end;

procedure TForm1.PaintBox1Paint(Sender: TObject);
var i: Integer;
begin
  FTreeDisplayer.Draw;
  PaintBox1.Width := FTreeDisplayer.DrawWidth;
  PaintBox1.Height := FTreeDisplayer.DrawHeight;
  {FXLeft := 0;
  FMaxWidth := 0;
  Memo1.Clear;
  FDataStru.TraceTree(nil, A0PreProcess, A0PostProcess);
  PaintBox1.Width := FXLeft;}
end;

procedure TForm1.PaintBox1MouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  FTreeDisplayer.SelectXY(X, Y);
end;

procedure TForm1.ActAddCriteriaExecute(Sender: TObject);
begin
  if FTreeDisplayer.SelCriteria = nil then begin
    SetToRoot(nil);
  end;
  DialogData.Edit2.Text := '';
  DialogData.Memo1.Clear;
  if DialogData.ShowModal = mrOK then begin
    FDataStru.AddCriteria(FTreeDisplayer.SelCriteria,
DialogData.Edit2.Text,
```

```
      DialogData.Memo1.Text);
    PaintBox1.Refresh;
  end;
  CalculateData;
  PaintBox1.Refresh;
end;

procedure TForm1.ActDeleteCriteriaExecute(Sender: TObject);
begin
  if FTreeDisplayer.SelCriteria <> nil then begin
    if MessageDlg('Delete Criteria ?', mtConfirmation, mbOKCancel,
0) = mrOK then begin
      FDataStru.DeleteCriteria(FTreeDisplayer.SelCriteria);
      FTreeDisplayer.SelCriteria := nil;
      CalculateData;
      PaintBox1.Refresh;
    end;
  end;
end;

procedure TForm1.ActModifyCriteriaExecute(Sender: TObject);
var vName, vDesc: string;
begin
  if FTreeDisplayer.SelCriteria <> nil then begin
    FDataStru.GetCritInfo(FTreeDisplayer.SelCriteria, vName, vDesc);
    DialogData.Edit2.Text := vName;
    DialogData.Memo1.Text := vDesc;
    if DialogData.ShowModal = mrOK then begin
      FDataStru.ModifyCriteria(FTreeDisplayer.SelCriteria,
DialogData.Edit2.Text,
        DialogData.Memo1.Text);
      PaintBox1.Refresh;
    end;
  end;
end;

procedure TForm1.ActAddAltExecute(Sender: TObject);
var vName, vDesc: string;
begin
  DialogData.Edit2.Text := '';
  DialogData.Memo1.Text := '';
  if DialogData.ShowModal = mrOK then begin
    vName := DialogData.Edit2.Text;
    vDesc := DialogData.Memo1.Text;
    if FDataStru.AltList.FieldByName(vName)<>nil then begin
      raise Exception.Create('Data Sudah Ada');
    end;
    FDataStru.AddAlternative(vName, vDesc);
  end;
  CalculateData;
  PaintBox1.Refresh;
end;
```

```
procedure TForm1.ActDeleteAltExecute(Sender: TObject);
begin
  if FTreeDisplayer.SelAlternative <> nil then begin
    if MessageDlg('Delete Criteria ?', mtConfirmation, mbOKCancel,
0) = mrOK then begin
      FDataStru.DeleteAlternative(FTreeDisplayer.SelAlternative);
      FTreeDisplayer.SelAlternative := nil;
      CalculateData;
      PaintBox1.Refresh;
    end;
  end;
end;

procedure TForm1.ActModifyAltExecute(Sender: TObject);
var vName, vDesc: string;
begin
  if FTreeDisplayer.SelAlternative <> nil then begin
    FDataStru.GetAltInfo(FTreeDisplayer.SelAlternative, vName,
vDesc);
    DialogData.Edit2.Text := vName;
    DialogData.Memo1.Text := vDesc;
    if DialogData.ShowModal=mrOK then begin
      vName := DialogData.Edit2.Text;
      vDesc := DialogData.Memo1.Text;
      FDataStru.ModifyAlternative(FTreeDisplayer.SelAlternative,
vName, vDesc);
    end;
  end;
end;

procedure TForm1.SetNextCrit(Sender: TObject);
var vBlock: TKBlockObject;
begin
  vBlock := FDataStru.GetNextCriteria(FTreeDisplayer.SelCriteria,
FAllowLeaf);
  if FTreeDisplayer.SelCriteria <> vBLock then begin
    FTreeDisplayer.SelCriteria := vBlock;
    PrepareProcess;
    PaintBox1.Refresh;
  end;
end;

procedure TForm1.SetPrevCrit(Sender: TObject);
var vBlock: TKBLockObject;
begin
  vBlock := FDataStru.GetPrevCriteria(FTreeDisplayer.SelCriteria,
FAllowLeaf);
  if FTreeDisplayer.SelCriteria <> vBLock then begin
    FTreeDisplayer.SelCriteria := vBlock;
    PrepareProcess;
    PaintBox1.Refresh;
  end;
end;
```

```
procedure TForm1.ActPreferenceExecute(Sender: TObject);
begin
  if FormPilihan.ShowModal = mrOK then begin
    FAllowLeaf := True;
    SetToRoot(nil);
    if FormPilihan.RadioInteger.Checked then begin
      FormQuest2.Status := 0;
    end else if FormPilihan.RadioReal.Checked then begin
      FormQuest2.Status := 1;
    end else begin
      FormQuest2.Status := 2;
    end;
    FWindowType := 1;
    FormQuest2.BtnPrev.OnClick := SetPrevCrit;
    FormQuest2.BtnNext.OnClick := SetNextCrit;
    PrepareProcess;
    FormQuest2.ShowModal;
    CalculateData;
    PaintBox1.Refresh;
    FWIndowType := 0;
  end;
end;

procedure TForm1.PrepareProcess;
begin
  case FWindowType of
    1: begin
      FormQuest2.DataStru := FDataStru;
      FormQuest2.CurrentCrit := FTreeDisplayer.SelCriteria;
      FormQuest2.BuildEntry;
    end;
    2: begin
      FormInterDependency.CritParent := FTreeDisplayer.SelCriteria;
    end;
  end;
end;

procedure TForm1.ActIndepExecute(Sender: TObject);
begin
  FAllowLeaf := False;
  SetToRoot(nil);
  FWindowType := 2;
  FormInterDependency.BtnPrev.OnClick := SetPrevCrit;
  FormInterDependency.BtnNext.OnClick := SetNextCrit;
  PrepareProcess;
  FormInterDependency.ShowModal;
  CalculateData;
  FWindowType := 0;
end;
```

```
procedure TForm1.CalculateData;
var vCrit: TKBlockObject;
  xCrit, vAlt: TKBlockObject;
  i, j: Integer;
  var vJml: Extended;
begin
  FDataStru.CalcAllCriteria;
  vCrit := FDataStru.Root;
  xCrit := nil;
  while xCrit <> vCrit do begin
    xCrit := vCrit;
    FormInterDependency.CritParent := vCrit;
    vCrit := FDataStru.GetNextCriteria(vCrit, False);
  end;
  for i:=0 to FDataStru.AltList.FieldCount-1 do begin
    try
    vAlt := TKBlockField(FDataStru.AltList[i]).BlockData;
    vAlt.TempReal1 := 0;
    except
    end;
  end;
  FDataStru.HitungAllAlternative;
  {for i:=0 to FDataStru.CritList.Count-1 do begin
    vCrit := FDataStru.CritList[i];
    if vCrit.FieldCount = 0 then begin
      for j:=0 to FDataStru.AltList.FieldCount-1 do begin
        vAlt := TKBlockField(FDataStru.AltList[j]).BlockData;
        vAlt.TempReal1 := vAlt.TempReal1+vCrit.TempReal3;
      end;
    end;
  end;
  vJml := 0;
  for i:=0 to FDataStru.AltList.FieldCount-1 do begin
    vAlt := TKBlockField(FDataStru.AltList[i]).BlockData;
    vJml := vJml+vAlt.TempReal1;
  end;
  for i:=0 to FDataStru.AltList.FieldCount-1 do begin
    vAlt := TKBlockField(FDataStru.AltList[i]).BlockData;
    vAlt.TempReal1 := vAlt.TempReal1/vJml;
  end;}
end;

procedure TForm1.FormShow(Sender: TObject);
begin
  CalculateData;
end;

procedure TForm1.ActNewFileExecute(Sender: TObject);
begin
  FormDocInfo.Edit1.Text := '';
  if FormDocInfo.ShowModal = mrOK then begin
    FDataStru.NewData;
    FDataStru.DocInfo := FormDocInfo.Edit1.Text;
    FTreeDisplayer.SelCriteria := nil;
    FTreeDisplayer.SelAlternative := nil;
    PaintBox1.Refresh;
```

```
    Panel1.Caption := FDataStru.DocInfo;
  end;
end;

procedure TForm1.ActLoadFileExecute(Sender: TObject);
begin
  if OpenDialog1.Execute then begin
    FDataStru.StruData.LoadFromFile(OpenDialog1.FileName);
    FDataStru.SynchronizeData;
    CalculateData;
    PaintBox1.Refresh;
    PaintBox1.Refresh;
    Panel1.Caption := FDataStru.DocInfo;
  end;
end;

procedure TForm1.ActSaveExecute(Sender: TObject);
begin
  if FileName = '' then begin
    ActSaveAsExecute(nil);
  end else begin
    FDataStru.StruData.SaveToFile(FileName);
    PaintBox1.Refresh;
  end;
end;

procedure TForm1.ActSaveAsExecute(Sender: TObject);
begin
  if SaveDialog1.Execute then begin
    FileName := SaveDialog1.FileName;
    ActSaveExecute(nil);
  end;
end;

procedure TForm1.ActSensExecute(Sender: TObject);
begin
  FormSensitivitas.Selected := nil;
  FormSensitivitas.SetCritParent(FDataStru);
  FormSensitivitas.RefreshCriteria := RefreshDataClick;
  FormSensitivitas.ShowModal;
end;

procedure TForm1.RefreshDataClick(Sender: TObject);
begin
  CalculateData;
end;

procedure TForm1.ActGSensExecute(Sender: TObject);
begin
  FormSensGraph.SetCritParent(FDataStru);
  FormSensGraph.ShowModal;
end;
```

```
procedure TForm1.Exit1Click(Sender: TObject);
begin
  Close;
end;

procedure TForm1.ActProgLinExecute(Sender: TObject);
begin
  FormProgramaLinier.SetDataStru(FDataStru);
  FormProgramaLinier.ShowModal;
end;

procedure TForm1.ActDocInfoExecute(Sender: TObject);
begin
  FormDocInfo.Edit1.Text := FDataStru.DocInfo;
  if (FormDocInfo.ShowModal = mrOK) then begin
    FDataStru.DocInfo := FormDocInfo.Edit1.Text;
    Panel1.Caption := FDataStru.DocInfo;
  end;
end;

procedure TForm1.ActPrintSetupExecute(Sender: TObject);
begin
  FormPrintHirarki.QuickRep1.PrinterSetup;
end;

procedure TForm1.ActCetakHierExecute(Sender: TObject);
begin
  FormPrintHirarki.DataStru := FDataStru;
  FormPrintHirarki.QRJudul.Caption := FDataStru.DocInfo;
  FormPrintHirarki.QRFooter.Caption := 'Hierarchy Report';
  FormPrintHirarki.Cetak(0);
end;

procedure TForm1.ActCetakHierListExecute(Sender: TObject);
begin
  FormPrintHirarki.DataStru := FDataStru;
  FormPrintHirarki.QRJudul.Caption := FDataStru.DocInfo;
  FormPrintHirarki.QRFooter.Caption := 'Hierarchy List';
  FormPrintHirarki.Cetak(1);
end;

procedure TForm1.SetupHierarchy;
begin
  ActPreference.Visible := False;
  ActIndep.Visible := False;
  ActSens.Visible := False;
  ActGSens.Visible := False;
  ToolButton8.Visible := False;
  ActProgLin.Visible := False;
  ActJudgement.Visible := False;
  ActReportWeight.Visible := False;
  ActReportIndep.Visible := False;
  ActReportSens.Visible := False;
  ActReportProglin.Visible := False;
  FTreeDisplayer.DrawValue := True;
end;
```

```
procedure TForm1.SetupGroup;
begin
  //
  ActGSens.Visible := False;
  ToolButton8.Visible := False;
  ActProgLin.Visible := False;
  ActJudgement.Visible := False;
  ActReportWeight.Visible := False;
  ActReportIndep.Visible := False;
  ActReportSens.Visible := False;
  ActReportProglin.Visible := False;
end;

procedure TForm1.SetupQuestionnaire;
begin
  //
  ActGSens.Visible := False;
  ToolButton8.Visible := False;
  ActProgLin.Visible := False;
  ActJudgement.Visible := False;
end;

procedure TForm1.SetupIndividual;
begin
  //Nope
end;

end.
```

```
unit UDataStru;

interface

uses
  SysUtils, Windows, Classes, KFieldObjects, USubject, UMsgList,
UDefault,
  GlobalUnit;

const
  // Message from DataStructure
  DSM_UPDATE = DSM_START +1;

type
  TTraceAction = procedure(vCrit: TKBlockObject; vLevel: Integer) of
object;
  TPrePosAction = procedure(vCrit: TKBlockObject; vLevel: Integer)
of object;
  TRDataStru = class(TRSubject)
  private
    FData: TKBlockObject;
    FCritList: TList;
    FAltList: TKBlockObject;
    FDefaultCriteria: TKFieldObject;
    FDefaultAlternative: TKFieldObject;
    FLevel: Integer;
    FTmpNameList: TStrings;
    FDocInfo: TKStringField;
    procedure AddCritToList(vCrit: TKBlockObject; vLevel: Integer);
    procedure DoCalcPreference(vCrit: TKBlockObject; vLevel:
Integer);
    function GetRoot: TKBlockObject;
    function GetDocInfo: string;
    procedure SetDocInfo(const Value: string);
  public
    constructor Create;
    property StruData: TKBlockObject read FData;
    destructor Destroy; override;
    procedure SaveToFile(FileName: string);
    property DefaultCriteria: TKFieldObject read FDefaultCriteria;
    property DefaultAlternative: TKFieldObject read
FDefaultAlternative;
    property AltList: TKBlockObject read FAltList;
    property CritList: TList read FCritList;
    property Root: TKBlockObject read GetRoot;
    procedure GetCritInfo(Current: TKBLockObject; var vName, vDesc:
string);
    procedure GetAltInfo(Current: TKFieldObject; var vName, vDesc:
string);
    property DocInfo: string read GetDocInfo write SetDocInfo;
    function FindCriteria(Name: string): Integer;
    function AddCriteria(Parent: TKBlockObject; CritName, CritDesc:
string): TKBlockObject;
    function DeleteCriteria(Current: TKBLockObject): Integer;
    function ModifyCriteria(Current: TKBlockObject; CritName,
CritDesc: string): Integer;
```

```
    function AddAlternative(AltName, AltDesc: string): TKBlockField;
    function DeleteAlternative(Current: TKFieldObject): Integer;
    function ModifyAlternative(Current: TKFieldObject; AltName,
AltDesc: string): Integer;
    function NewData: Integer;
    procedure SynchronizeData;
    procedure TraceCriteria(Parent: TKBlockObject; NodeAction:
TTraceAction;
      StartAction, EndAction: TPrePosAction; Recurse: Boolean);
    procedure TraceTree(NodeAction: TTraceAction; StartAction,
EndAction: TPrePosAction;
      Recurse: Boolean);
    function GetNextCriteria(vBlock: TKBlockObject; vAllowLeaf:
Boolean): TKBlockOBject;
    function GetPrevCriteria(vBlock: TKBlockObject; vAllowLeaf:
Boolean): TKBlockObject;
    procedure SetMatrixData(vBlock: TKBlockObject; vId: string;
vTanda: Integer; vNilai: Extended);
    procedure GetMatrixData(vBlock: TKBlockObject; vId: string; var
vTanda: Integer; var vNilai: Extended);
    procedure HitungKriteria(vBlock: TKBlockObject;
      var FTemporerArray: TArrayCriteria);
    procedure HitungAlternative(vBlock: TKBlockObject;
      var FTemporerArray: TArrayCriteria);
    procedure CalcAllCriteria;
    procedure HitungAllAlternative;
  end;

implementation

{ TRDataStru }
var TMPArray: TArrayCriteria;

function TRDataStru.AddAlternative(AltName, AltDesc: string):
TKBlockField;
var vField: TKFieldObject;
begin
  result := nil;
  if FAltList.FieldByName(AltName)<>nil then begin
    raise Exception.Create('Alternative Sudah Ada');
  end else begin
    result := TKBlockField(FAltList.AddField(CFIELD_BLOCK,
AltName));

result.BlockData.Assign(TKBlockField(FDefaultAlternative).BlockData)
;
    vField := result.BlockData.FieldByName('Description');
    if vField is TKStringField then begin
      TKStringField(vField).StringData := AltDesc;
    end;
  end;
end;
```

```
function TRDataStru.AddCriteria(Parent: TKBlockObject; CritName,
  CritDesc: string): TKBlockObject;
var vIndex: Integer;
  vField: TKFieldObject;
  xField: TKFieldObject;
  vBlock: TKBlockObject;
begin
  result := nil;
  vIndex := FindCriteria(CritName);
  if vIndex>=0 then begin
    raise Exception.Create('Data Sudah Ada');
  end;
  if Parent = nil then begin
    xField := FData.FieldByName('Criteria');
    vBlock := TKBlockField(xField).BlockData;
  end else begin
    vBlock := Parent;
    {if Parent is TKBlockField then begin
      if vIndex>=0 then begin
        raise Exception.Create('Data Sudah Ada');
      end else begin
        xField :=
TKBlockField(Parent).BlockData.FieldByName('Criteria');
        vBlock := TKBlockField(xField).BlockData;
      end;
    end else begin
      raise Exception.Create('Not Valid Parent');
    end;}
  end;
  if vBlock <> nil then begin
    xField := vBlock.AddField(CFIELD_BLOCK, CritName);
    TKBlockField(xField).BlockData.Assign(
      TKBlockField(DefaultCriteria).BlockData);
    vField :=
TKBlockField(xField).BlockData.FieldByName('Description');
    if vField is TKStringField then begin
      TKStringField(vField).StringData := CritDesc;
    end;

    result :=
TKBlockField(TKBlockField(xField).BlockData.FIeldByName('Criteria'))
.BlockData;
  end;
  SynchronizeData;
end;

procedure TRDataStru.AddCritToList(vCrit: TKBlockObject; vLevel:
Integer);
begin
  FCritList.Add(vCrit);
end;
```

```
procedure TRDataStru.CalcAllCriteria;
begin
{    procedure TraceTree(NodeAction: TTraceAction; StartAction,
EndAction: TPrePosAction;
      Recurse: Boolean);
}
  TraceTree(nil, DoCalcPreference, nil, True);
end;

constructor TRDataStru.Create;
begin
  inherited;
  FTmpNameList := TStringList.Create;
  FData := TKBlockObject.Create;
  FCritList := TList.Create;
  NewData;
end;

function TRDataStru.DeleteAlternative(Current: TKFieldObject):
Integer;
begin
  Current.Free;
  result := 0;
end;

function TRDataStru.DeleteCriteria(Current: TKBLockObject): Integer;
begin
  result := 0;
  if Current.OwnerField.Owner.OwnerField <> nil then begin
    Current.OwnerField.Owner.OwnerField.Free;
    SynchronizeData;
  end;
end;

destructor TRDataStru.Destroy;
begin
  FData.Free;
  FCritList.Free;
  FTmpNameList.Free;
  inherited;
end;

procedure TRDataStru.DoCalcPreference(vCrit: TKBlockObject;
  vLevel: Integer);
var i: Integer;
  vField: TKFieldObject;
  vParentValue: Extended;
begin
  if vCrit.OwnerField.Owner.OwnerField = nil then begin
    vParentValue := 1;
    vCrit.TempReal1 := 1;
    vCrit.TempReal2 := 1;
    vCrit.TempReal3 := 1;
    vCrit.TempReal4 := 1;
  end else begin
    vParentValue := vCrit.TempReal1;
```

```
  end;
  HitungKriteria(vCrit, TMPArray);
  if vCrit.FieldCount<>0 then begin
    for i:=0 to vCrit.FieldCount-1 do begin
      vField := vCrit[i];
      vField :=
TKBlockField(vField).BlockData.FieldByName('Criteria');
      TKBlockField(vFIeld).BlockData.TempReal2 :=
TMPArray[vCrit.FieldCount+1, i];
      TKBlockField(vFIeld).BlockData.TempReal1 :=
vParentValue*TMPArray[vCrit.FieldCount+1, i];
      TKBlockField(vFIeld).BlockData.TempReal4 :=
TKBlockField(vFIeld).BlockData.TempReal2;
      TKBlockField(vFIeld).BlockData.TempReal3 :=
TKBlockField(vFIeld).BlockData.TempReal1;
    end;
  end;
end;

function TRDataStru.FindCriteria(Name: string): Integer;
var i: Integer;
  vObj: TKFieldObject;
  vBlock: TKBlockObject;
  vName: string;
begin
  result := -1;
  for i:=0 to FCritList.Count-1 do begin
    vBlock := FCritList[i];
    if TKBlockField(vBlock.OwnerField).Owner.OwnerField = nil then
begin
      vName := 'Goal';
    end else begin
      vName :=
TKBlockField(vBlock.OwnerField).Owner.OwnerField.FieldName;
    end;
    if CompareText(vName, Name) = 0 then begin
      result := i;
      exit;
    end;
  end;
end;

procedure TRDataStru.GetAltInfo(Current: TKFieldObject; var vName,
  vDesc: string);
var vFIeld: TKFieldObject;
begin
  if Current = nil then exit;
  vName := Current.FieldName;
  vField :=
TKBlockField(Current).BlockData.FieldByName('Description');
  if vField is TKStringFIeld then begin
    vDesc := TKStringField(vFIeld).StringData
  end;
end;
```

```
procedure TRDataStru.GetCritInfo(Current: TKBLockObject; var vName,
  vDesc: string);
var vBlock: TKBlockObject;
  vField: TKFieldObject;
begin
  try
    if Current.OwnerField.Owner<>nil then begin
      vBlock := Current.OwnerField.Owner;
      if vBlock.OwnerField.Owner.OwnerField <> nil then begin
        vName := Current.OwnerField.Owner.OwnerField.FieldName;
      end else begin
        vName := 'Goal';
      end;
      vFIeld := vBlock.FieldByName('Description');
      if vField is TKStringField then begin
        vDesc := TKStringField(vField).StringData;
      end;
    end;
  except
  end;
end;

procedure TRDataStru.GetMatrixData(vBlock: TKBlockObject; vId:
string;
  var vTanda: Integer; var vNilai: Extended);
var xBlock: TKBlockObject;
  vField: TKFieldObject;
begin
  try
    xBlock := vBlock.OwnerField.Owner;
    vField := xBLock.FieldByName('Matrix');
    vBlock := TKBlockField(vField).BlockData;
    vFIeld := vBlock.FIeldByName(vId);
    if vField is TKFloatField then begin
      vNilai := TKFloatField(vField).FloatData;
      vTanda := vField.TagValue1;
      if vNilai<1 then vNilai := 1;
      if vNilai>9 then vNilai := 9;
    end else begin
      vTanda := 0;
      vNilai := 1;
    end;
  except
  end;
end;

function TRDataStru.GetNextCriteria(vBlock: TKBlockObject;
vAllowLeaf: Boolean): TKBlockObject;
var vIndex: Integer;
  xBlock: TKBlockObject;
begin
  if vBlock=nil then begin
    result := nil;
  end else begin
    vIndex := CritList.IndexOf(vBlock);
    if vIndex>=0 then begin
```

```
      if vIndex<CritList.Count-1 then begin
        if vAllowLeaf then begin
          result := CritList[vIndex+1];
        end else begin
          result := nil;
          xBlock := CritList[vIndex+1];
          while (xBlock.FieldCount=0) and (vIndex<CritList.Count-1)
do begin
            xBlock := CritList[vIndex+1];
            inc(vIndex);
          end;
          if xBlock.FieldCount>0 then begin
            result := xBlock;
          end else begin
            result := vBlock;
          end;
        end;
      end else begin
        result := vBlock;
      end;
    end else begin
      result := vBlock;
    end;
  end;
end;

function  TRDataStru.GetPrevCriteria(vBlock: TKBlockObject;
vAllowLeaf: Boolean): TKBlockObject;
var vIndex: Integer;
  xBlock: TKBlockObject;
begin
  if vBlock=nil then begin
    result := nil;
  end else begin
    vIndex := CritList.IndexOf(vBlock);
    if vIndex>0 then begin
      if vAllowLeaf then begin
        result := CritList[vIndex-1];
      end else begin
        xBlock := CritList[vIndex-1];
        while (xBlock.FieldCount=0) and (vIndex>0) do begin
          xBlock := CritList[vIndex-1];
          dec(vIndex);
        end;
        if xBlock.FieldCount>0 then begin
          result := xBlock;
        end else begin
          result := vBlock;
        end;
      end;
    end else begin
      result := vBlock;
    end;
  end;
end;
```

```
function TRDataStru.GetRoot: TKBlockObject;
var vField: TKFieldObject;
begin
  result := nil;
  vField := FData.FieldByName('Criteria');
  if vFIeld is TKBlockField then begin
    result := TKBlockField(vFIeld).BlockData;
  end;
end;

procedure TRDataStru.HitungAllAlternative;
var i: Integer;
  vBlock: TKBlockObject;
  vArr: TArrayCriteria;
  j: Integer;
  vJmlAlt: Extended;
begin
  for i:=0 to FAltList.FieldCount-1 do begin
    TKBlockField(FAltList[i]).BlockData.TempReal1 := 0;
  end;
  for i:=0 to FCritList.Count-1 do begin
    vBlock := FCritList[i];
    if vBlock.FieldCount=0 then begin
      HitungAlternative(vBlock, vArr);
      for j:=0 to FAltList.FieldCount-1 do begin
        TKBlockField(FAltList[j]).BlockData.TempReal1 :=
          TKBlockField(FAltList[j]).BlockData.TempReal1+
          vArr[FAltList.FieldCount+1, j]*vBlock.TempReal3;
      end;
    end;
  end;
  vJmlAlt := 0;
  for j:=0 to FAltList.FieldCount-1 do begin
    vJmlAlt := vJmlAlt +
TKBlockField(FAltList[j]).BlockData.TempReal1;
  end;
  for j:=0 to FAltList.FieldCount-1 do begin
    TKBlockField(FAltList[j]).BlockData.TempReal1 :=
      TKBlockField(FAltList[j]).BlockData.TempReal1/vJmlAlt;
  end;
end;

procedure TRDataStru.HitungKriteria(vBlock: TKBlockObject;
  var FTemporerArray: TArrayCriteria);
var i, j: Integer;
  vTanda: Integer;
  vNilai: Extended;
  vId: string;
  vTotal: Extended;
  vSum: Extended;
begin
  FTmpNameList.Clear;
  for i:=0 to vBlock.FieldCount-1 do begin
    FTmpNameList.Add(vBlock[i].FieldName);
  end;
  for i:=0 to FTmpNameList.Count-1 do begin
```

```
    for j:=0 to FTmpNameList.Count-1 do begin
      if i=j then begin
        FTemporerArray[i, j] := 1;
      end else if i>j then begin
        vId := FTmpNameList[j]+','+FTmpNameList[i];
        GetMatrixData(vBlock, vId, vTanda, vNilai);
        if vTanda = 0 then begin
          FTemporerArray[i, j] := 1/vNilai;
        end else begin
          FTemporerArray[i, j] := vNilai;
        end;
      end else begin
        vId := FTmpNameList[i]+','+FTmpNameList[j];
        GetMatrixData(vBlock, vId, vTanda, vNilai);
        if vTanda = 0 then begin
          FTemporerArray[i, j] := vNilai;
        end else begin
          FTemporerArray[i, j] := 1/vNilai;
        end;
      end;
    end;
  end;
  vSum := 0;
  for j:=0 to FTmpNameList.Count-1 do begin
    vTotal := 1;
    for i:=0 to FTmpNameList.Count-1 do begin
      vTotal := vTotal*FTemporerArray[i, j];
    end;
    FTemporerArray[FTmpNameList.Count, j] := HitungAkar(vTotal,
FTmpNameList.Count);
    vSum := vSum+FTemporerArray[FTmpNameList.Count, j];
  end;
  for j:=0 to FTmpNameList.Count-1 do begin
    FTemporerArray[FTmpNameList.Count+1, j] :=
FTemporerArray[FTmpNameList.Count, j]/vSum;
  end;
end;

procedure TRDataStru.HitungAlternative(vBlock: TKBlockObject;
  var FTemporerArray: TArrayCriteria);
var i, j: Integer;
  vTanda: Integer;
  vNilai: Extended;
  vId: string;
  vTotal: Extended;
  vSum: Extended;
begin
  if vBlock.FieldCount <> 0 then exit;
  FTmpNameList.Clear;
  for i:=0 to AltList.FieldCount-1 do begin
    FTmpNameList.Add(AltList[i].FieldName);
  end;
  for i:=0 to FTmpNameList.Count-1 do begin
    for j:=0 to FTmpNameList.Count-1 do begin
      if i=j then begin
        FTemporerArray[i, j] := 1;
```

```
      end else if i>j then begin
        vId := FTmpNameList[j]+','+FTmpNameList[i];
        GetMatrixData(vBlock, vId, vTanda, vNilai);
        if vTanda = 0 then begin
          FTemporerArray[i, j] := 1/vNilai;
        end else begin
          FTemporerArray[i, j] := vNilai;
        end;
      end else begin
        vId := FTmpNameList[i]+','+FTmpNameList[j];
        GetMatrixData(vBlock, vId, vTanda, vNilai);
        if vTanda = 0 then begin
          FTemporerArray[i, j] := vNilai;
        end else begin
          FTemporerArray[i, j] := 1/vNilai;
        end;
      end;
    end;
  end;
  vSum := 0;
  for j:=0 to FTmpNameList.Count-1 do begin
    vTotal := 1;
    for i:=0 to FTmpNameList.Count-1 do begin
      vTotal := vTotal*FTemporerArray[i, j];
    end;
    FTemporerArray[FTmpNameList.Count, j] := HitungAkar(vTotal,
FTmpNameList.Count);
    vSum := vSum+FTemporerArray[FTmpNameList.Count, j];
  end;
  for j:=0 to FTmpNameList.Count-1 do begin
    FTemporerArray[FTmpNameList.Count+1, j] :=
FTemporerArray[FTmpNameList.Count, j]/vSum;
  end;
end;

function TRDataStru.ModifyAlternative(Current: TKFieldObject;
AltName,
  AltDesc: string): Integer;
var vField: TKFieldObject;
begin
  result := 0;
  if CompareText(Current.FieldName, AltName)<>0 then begin
    if FAltList.FieldByName(AltName)=nil then begin
      Current.SetName(AltName);
    end else begin
      raise Exception.Create('Data Sudah Ada');
    end;
  end;
  vField :=
TKBlockField(Current).BlockData.FieldByName('Description');
  if vField is TKStringFIeld then begin
    TKStringField(vFIeld).StringData := AltDesc;
  end;
end;
```

```
function TRDataStru.ModifyCriteria(Current: TKBLockObject; CritName,
  CritDesc: string): Integer;
var vName: string;
  vField: TKFieldObject;
  vBlock: TKBLockObject;
begin
  result := 0;
  if Current.OwnerField.Owner<>nil then begin
    vBlock := Current.OwnerField.Owner;
    if Current.OwnerField.Owner.OwnerField <> nil then begin
      vName := Current.OwnerField.Owner.OwnerField.FieldName;
      if CompareText(vName, CritName)<>0 then begin
        if FindCriteria(CritName)>=0 then begin
          raise Exception.Create('Data Sudah Ada');
        end;
        Current.OwnerField.Owner.OwnerField.SetName(CritName);
      end;
    end;
    vFIeld := vBlock.FieldByName('Description');
    if vField is TKStringField then begin
      TKStringField(vField).StringData := CritDesc;
    end;
    SynchronizeData;
  end;
end;

function TRDataStru.NewData: Integer;
begin
  try
    FCritList.Clear;
    FData.ClearFields;
    DefaultStream.Position := 0;
    FData.LoadFromStream(DefaultStream);
    SynchronizeData;
    Notify(DSM_UPDATE, 0, 0);
  except
    result := -1;
    exit;
  end;
  result := 0;
end;

procedure TRDataStru.SaveToFile(FileName: string);
begin
  FData.SaveToFile(FileName);
end;

procedure TRDataStru.SetMatrixData(vBlock: TKBlockObject; vId:
string;
  vTanda: Integer; vNilai: Extended);
var xBlock: TKBlockObject;
  vField: TKFieldObject;
begin
  try
    xBlock := vBlock.OwnerField.Owner;
    vField := xBLock.FieldByName('Matrix');
```

```
    vBlock := TKBlockField(vField).BlockData;
    vFIeld := vBlock.FIeldByName(vId);
    if vField is TKFloatField then begin
      TKFloatField(vField).FloatData := vNilai;
      vField.TagValue1 := vTanda;
      if vNilai<1 then vNilai := 1;
      if vNilai>9 then vNilai := 9;
    end else begin
      if vFIeld = nil then begin
        vField := vBlock.AddField(CFIELD_FLOAT, vId);
      end else begin
        vFIeld.Free;
        vField := vBlock.AddField(CFIELD_FLOAT, vId);
      end;
      TKFloatField(vField).FloatData := vNilai;
      vField.TagValue1 := vTanda;
    end;
  except
  end;
end;

procedure TRDataStru.SynchronizeData;
var vField: TKFieldObject;
begin
  FCritList.Clear;
  TraceTree(nil, AddCritToList, nil, True);
  vField := FData.FieldByName('Alternative');
  if vField is TKBlockField then begin
    FAltList := TKBlockField(vField).BlockData;
  end else begin
    raise Exception.Create('Data Error');
  end;
  vField := FData.FieldByName('DefaultCriteria');
  if vField is TKBlockField then begin
    FDefaultCriteria := vField;
  end else begin
    raise Exception.Create('Data Error');
  end;
  vField := FData.FieldByName('DefaultAlternative');
  if vField is TKBlockField then begin
    FDefaultAlternative := vField;
  end else begin
    raise Exception.Create('Data Error');
  end;
  vField := FData.FieldByName('DocInfo');
  if vField = nil then begin
    vField := FData.AddField(CFIELD_STRING, 'DocInfo');
  end;
  if vField is TKStringField then begin
    FDocInfo := TKStringField(vField);
  end else begin
    raise Exception.Create('Data Error');
  end;
  CalcAllCriteria;
end;
```

```
procedure TRDataStru.TraceCriteria(Parent: TKBlockObject;
NodeAction: TTraceAction;
      StartAction, EndAction: TPrePosAction; Recurse: Boolean);
var vBlock: TKBlockObject;
  vField: TKFieldObject;
  xField: TKFieldObject;
  i: Integer;
begin
  if Assigned(StartAction) then begin
    StartAction(Parent, FLevel);
  end;
  FLevel := FLevel+1;
  for i:=0 to Parent.FieldCount-1 do begin
    vField := Parent[i];
    if vField is TKBlockField then begin
      xField :=
TKBlockField(vField).BlockData.FieldByName('Criteria');
      if xField is TKBlockField then begin
        vBlock := TKBlockField(xField).BlockData;
        if Assigned(NodeAction) then begin
          NodeAction(vBlock, FLevel);
        end;
        if Recurse then begin
        TraceCriteria(vBlock, NodeAction, StartAction, EndAction,
Recurse);
        end;
      end;
    end;
  end;
  FLevel := FLevel-1;
  if Assigned(EndAction) then begin
    EndAction(Parent, FLevel);
  end;
end;

procedure TRDataStru.TraceTree(NodeAction: TTraceAction;
StartAction,
  EndAction: TPrePosAction; Recurse: Boolean);
var vField: TKFieldObject;
begin
  FLevel := 0;
  vField := FData.FieldByName('Criteria');
  if vField is TKBlockField then begin
    TraceCriteria(TKBlockField(vField).BlockData, NodeAction,
StartAction, EndAction, Recurse);
  end;
end;
```

```
function TRDataStru.GetDocInfo: string;
begin
  result := '';
  try
    result := FDocInfo.StringData;
  except
  end;
end;

procedure TRDataStru.SetDocInfo(const Value: string);
begin
  try
    FDocInfo.StringData := Value;
  except
  end;
end;

end.
```

```
unit USubject;

interface
uses
  SysUtils, Classes, Messages;

type
  TRSubject = class
  private
    FObserverList: TList;
  public
    constructor Create;
    destructor Destroy; override;
    procedure RegisterObserver(vObj: TObject);
    procedure UnRegisterObserver(vObj: TObject);
    function Notify(MsgID, WParam, LParam: Longint): Integer;
  end;

implementation

{ TR3Subject }

constructor TRSubject.Create;
begin
  FObserverList := TList.Create;
end;

destructor TRSubject.Destroy;
begin
  FObserverList.Free;
  inherited;
end;

function TRSubject.Notify(MsgID, WParam, LParam: Integer): Integer;
var vMSG: TMessage;
  vObj: TObject;
  i: Integer;
begin
  vMSG.Msg := MsgID;
  vMsg.WParam := WParam;
  vMsg.LParam := LParam;
  for i:=0 to FObserverList.Count-1 do begin
    vObj := FObserverList[i];
    vObj.Dispatch(vMsg);
  end;
  result := 0;
end;

procedure TRSubject.RegisterObserver(vObj: TObject);
begin
  if FObserverList.IndexOf(vObj)<0 then begin
    FObserverList.Add(vObj);
  end;
end;
```

```
procedure TRSubject.UnRegisterObserver(vObj: TObject);
var vIndex: Integer;
begin
  vIndex := FObserverList.IndexOf(vObj);
  if vIndex >=0 then begin
    FObserverList.Delete(vIndex);
  end;
end;

end.
```

```
unit UMsgList;

interface

uses Messages;

const // Message ID Cluster
  DSM_START = WM_USER+0;  // Cluster message for Data Structure

implementation

end.
```

```
unit KFieldObjects;

interface

uses
  Windows, Messages, SysUtils, Classes;

const
  CBLOCKID = $2042534B;
  CDATAID  = 180797;

  CFIELD_INTEGER      = $01000000;
  CFIELD_FLOAT        = $02000000;
  CFIELD_STRING       = $03000000;
  CFIELD_MEMO         = $04000000;
  CFIELD_BINARY       = $05000000;
  CFIELD_GRAPHIC      = $06000000;
  CFIELD_BLOCK        = $07000000;
  CFIELD_INDEX        = $08000000;
  CFIELD_STRINGINDEX  = $09000000;
  CFIELD_STRINGPATH   = $0A000000;

  CVERMAYOR      = 1;
  CVERMINOR      = 0;

  CInvalidBlockData = 'Invalid Block Data';

type
  {$A-}
  TKFieldName = string[63];
  TKBlockHeader = record
    BlockId: Longint;
    DataId: Longint;
    ObjCount: Longint;
    ObjSize: Longint;
    VerMayor: Word;
    VerMinor: Word;
    BlockFileId: Longint;
    Reserved1: Longint;
    Reserved2: Longint;
  end;
  TKDataHeader = record
    DataID: Longint;
    FieldCode: Longint;
    DataSize: Longint;
    DataCount: Longint;
    {FieldName: TKFieldName;}
    NameLength: Longint;

    IntegerData: Longint;
    FloatData: Extended;
    DataFiler: Word;
    TagValue1: Longint;
    TagValue2: Longint;
    TagValue3: Longint;
    Reserved1: Longint;
```

```
    Reserved2: Longint;
    Reserved3: Word;
    Reserved4a: Byte;
    Reserved4b: Byte;
  end;
  {$A+}
  TKFieldObject = class;
  TKBlockObject = class;
  TKOldCreation = function(ClassId: Longint; Owner: TKBlockObject):
TKFieldObject;
  TKCreation = function(OldCreation: TKOldCreation; ClassId:
Longint;
    Owner: TKBlockObject): TKFieldObject;

  TKBlockObject = class(TPersistent)
  private
    FFieldList: TList;
    FOwnerField: TKFieldObject;
    FBlockFileId: Longint;
    FTempReal1: Double;
    FTempReal2: Double;
    FTempReal3: Double;
    FTempReal4: Double;
    FTempInteger1: Integer;
    FTempInteger2: Integer;
    procedure SetTempInteger1(const Value: Integer);
    procedure SetTempInteger2(const Value: Integer);
  protected
    procedure Assign(Source: TPersistent); override;
    procedure AssignTo(Dest: TPersistent); override;
    function GetFieldData(const Indeks: Integer): TKFieldObject;
    function GetFieldCount: Integer;
    procedure SaveDataToStream(Stream: TStream); virtual;
    procedure LoadDataFromStream(Stream: TStream); virtual;
  public
    property TempReal1: Double read FTempReal1 write FTempReal1;
    property TempReal2: Double read FTempReal2 write FTempReal2;
    property TempReal3: Double read FTempReal3 write FTempReal3;
    property TempReal4: Double read FTempReal4 write FTempReal4;
    property TempInteger1: Integer read FTempInteger1 write
SetTempInteger1;
    property TempInteger2: Integer read FTempInteger2 write
SetTempInteger2;
    function Modified: Boolean;
    property BlockFileId: Longint read FBlockFileId write
FBlockFileId;
    procedure SaveToStream(Stream: TStream); virtual;
    procedure LoadFromStream(Stream: TStream); virtual;
    function LoadFromFile(FileName: string): Integer;
    procedure SaveToFile(FileName: string);
    property OwnerField: TKFieldObject read FOwnerField;
    property FieldList: TList read FFieldList;
    property FieldCount: Integer read GetFieldCount;
    property FieldData[const Indeks: Integer]:TKFieldObject read
GetFieldData; default;
    function FieldByName(FieldName: TKFieldName): TKFieldObject;
```

```
    function AddField(ClassId: Longint; FieldName: TKFieldName):
TKFieldObject;
    procedure ClearFields;
    procedure RefreshBlockBinary;
    procedure RefreshBlockObject;
    procedure MoveUp(Index: Integer);
    procedure MoveDown(Index: Integer);
    function ChangeFieldName(CurrentName: string; NewName: string):
Boolean;
    constructor Create; virtual;
    Destructor Destroy; override;
  end;

  TKFieldObject = class(TPersistent)
  protected
    ClassId: Longint;
    FFieldName: TKFieldName;
    FOwner: TKBlockObject;
    FTagValue1: Longint;
    FTagValue2: Longint;
    FTagValue3: Longint;
    procedure FillDataHeader(var vDataHD: TKDataHeader); virtual;
    procedure LoadDataHeader(vDataHD: TKDataHeader); virtual;
    function GetDataSize: Longint; virtual;
    procedure WriteStream(Stream: TStream); virtual;
    procedure ReadStream(Stream: TStream; Size: Longint); virtual;
  public
    procedure SetName(vName: String);
    property IdClass: Longint read ClassId;
    property Owner: TKBLockObject read FOwner;
    property FieldName: TKFieldName read FFieldName;
    property TagValue1: Longint read FTagValue1 write FTagValue1;
    property TagValue2: Longint read FTagValue2 write FTagValue2;
    property TagValue3: Longint read FTagValue3 write FTagValue3;
    constructor Create(Owner: TKBlockObject); virtual;
    destructor Destroy; override;
  end;

  TKIntegerField = class(TKFieldObject)
  protected
    FIntegerData: Longint;
    procedure FillDataHeader(var vDataHD: TKDataHeader); override;
    procedure LoadDataHeader(vDataHD: TKDataHeader); override;
  public
    property IntegerData: Integer read FIntegerData write
FIntegerData;
    constructor Create(Owner: TKBlockObject); override;
    destructor Destroy; override;
  end;

  TKIndexField = class(TKIntegerField)
  end;
```

```
TKFloatField = class(TKFieldObject)
protected
  FFloatData: Extended;
  procedure FillDataHeader(var vDataHD: TKDataHeader); override;
  procedure LoadDataHeader(vDataHD: TKDataHeader); override;
public
  property FloatData: Extended read FFloatData write FFloatData;
  constructor Create(Owner: TKBlockObject); override;
  destructor Destroy; override;
end;

TKStringField = class(TKFieldObject)
protected
  FStringData: String;
  procedure FillDataHeader(var vDataHD: TKDataHeader); override;
  function GetDataSize: Longint; override;
  procedure WriteStream(Stream: TStream); override;
  procedure ReadStream(Stream: TStream; Size: Longint); override;
public
  property StringData: string read FStringData write FStringData;
  constructor Create(Owner: TKBlockObject); override;
  destructor Destroy; override;
end;

TKMemoField = class(TKStringField)
end;

TKStringIndexField = class(TKStringField)
end;

TKStringPathField = class(TKStringField)
end;

TKBinaryField = class(TKFieldObject)
protected
  FBinaryData: TMemoryStream;
  procedure FillDataHeader(var vDataHD: TKDataHeader); override;
  function GetDataSize: Longint; override;
  procedure WriteStream(Stream: TStream); override;
  procedure ReadStream(Stream: TStream; Size: Longint); override;
public
  property BinaryData: TMemoryStream read FBinaryData;
  procedure SaveToStream(Stream: TStream); virtual;
  procedure LoadFromStream(Stream: TStream); virtual;
  procedure SaveToFile(FileName: string); virtual;
  procedure LoadFromFile(FileName: string); virtual;
  constructor Create(Owner: TKBlockObject); override;
  destructor Destroy; override;
end;

TKGraphicField = class(TKBinaryField)
end;
```

```
  TKBlockField = class(TKBinaryField)
  protected
    FBlockData: TKBlockObject;
  public
    property BlockData: TKBlockObject read FBlockData;
    constructor Create(Owner: TKBlockObject); override;
    destructor Destroy; override;
  end;

var
  KFieldCreation: TKCreation;

function FieldIdToString(ClassId: Longint): string;

implementation

function FieldObjectCreation(ClassId: Longint; Owner:
TKBlockObject): TKFieldObject;
begin
  case ClassId of
    CFIELD_INTEGER: result := TKIntegerField.Create(Owner);
    CFIELD_FLOAT: result := TKFloatField.Create(Owner);
    CFIELD_STRING: result := TKStringField.Create(Owner);
    CFIELD_MEMO: result := TKMemoField.Create(Owner);
    CFIELD_BINARY: result := TKBinaryField.Create(Owner);
    CFIELD_GRAPHIC: result := TKGraphicField.Create(Owner);
    CFIELD_BLOCK: result := TKBlockField.Create(Owner);
    CFIELD_INDEX: result := TKIndexField.Create(Owner);
    CFIELD_STRINGINDEX: result := TKStringIndexField.Create(Owner);
    CFIELD_STRINGPATH: result := TKStringPathField.Create(Owner);
  else
    result := nil;
  end;
end;

function FieldIdToString(ClassId: Longint): string;
begin
  case ClassId of
    CFIELD_INTEGER: result := 'TKIntegerField';
    CFIELD_FLOAT: result := 'TKFloatField';
    CFIELD_STRING: result := 'TKStringField';
    CFIELD_MEMO: result := 'TKMemoField';
    CFIELD_BINARY: result := 'TKBinaryField';
    CFIELD_GRAPHIC: result := 'TKBinaryField';
    CFIELD_BLOCK: result := 'TKBlockField';
  else
    result := '';
  end;
end;
```

```
procedure WriteString(vStream: TStream; const vStr: string);
var vPtr: Pointer;
  vLen: Longint;
begin
  vLen := Length(vStr);
  vPtr := AllocMem(vLen+1);
  StrPLCopy(vPtr, vStr, vLen);
  vStream.Write(vPtr^, vLen);
  FreeMem(vPtr);
end;

procedure ReadString(vStream: TStream; var vStr: string; Len:
Longint);
var vPtr: Pointer;
begin
  vPtr := AllocMem(Len+1);
  vStream.Read(vPtr^, Len);
  vStr := StrPas(vPtr);
  FreeMem(vPtr);
end;

(* TKBlockObject *)

constructor TKBlockObject.Create;
begin
  inherited;
  FFieldList := TList.Create;
  FOwnerField := nil;
end;

destructor TKBlockObject.Destroy;
begin
  ClearFields;
  FFieldList.Destroy;
  inherited;
end;

procedure TKBlockObject.ClearFields;
var i: Integer;
begin
  for i:=FFieldList.Count-1 downto 0 do begin
    TKFieldObject(FFieldList[i]).Destroy;
  end;
  FFieldList.Clear;
end;

function TKBlockObject.GetFieldData(const Indeks: Integer):
TKFieldObject;
var vField: TKFieldObject;
begin
  result := nil;
  if (Indeks>=0) and (Indeks<FFieldList.Count) then begin
    vField := FFieldList[Indeks];
    result := vField;
  end;
end;
```

```
function TKBlockObject.GetFieldCount: Integer;
begin
  result := FFieldList.Count;
end;

procedure TKBlockObject.Assign(Source: TPersistent);
var vMemStream: TMemoryStream;
begin
  if Source is TKBlockObject then begin
    try
      vMemStream := TMemoryStream.Create;
      TKBlockObject(Source).RefreshBlockBinary;
      TKBlockObject(Source).SaveDataToStream(vMemStream);
      vMemStream.Position := 0;
      LoadDataFromStream(vMemStream);
      RefreshBlockObject;
      vMemStream.Destroy;
    except
    end;
  end else begin
    inherited;
  end;
end;

procedure TKBLockObject.AssignTo(Dest: TPersistent);
begin
  if Dest<>nil then begin
    if Dest is TKBlockObject then begin
      Dest.Assign(Self);
    end else begin
      inherited;
    end;
  end else begin
    inherited;
  end;
end;

function TKBlockObject.AddField(ClassId: Longint; FieldName:
TKFieldName): TKFieldObject;
begin
  if Assigned(KFieldCreation) then begin
    result := KFieldCreation(FieldObjectCreation, ClassId, Self);
    result.FFieldName := FieldName;
  end else begin
    result := FieldObjectCreation(ClassId, Self);
    if result <> nil then begin
      result.FFieldName := FieldName;
    end;
  end;
  if result <> nil then begin
    result.ClassId := ClassId;
  end;
end;
```

```
function TKBlockObject.ChangeFieldName(CurrentName: string; NewName:
string): Boolean;
var vField: TKFieldObject;
begin
  result := True;
  vField := FieldByName(CurrentName);
  if vField <> nil then begin
    vField.FFieldName := NewName;
  end else begin
    result := False;
  end;
end;

procedure InitBlockHD(var vBlockHD: TKBlockHeader);
begin
  FillChar(vBlockHd, SizeOf(vBlockHd), 0);
  vBlockHD.BlockId := CBLOCKID;
  vBlockHD.DataId := CDATAID;
  vBlockHD.ObjCount := 0;
  vBlockHD.ObjSize := 0;
  vBlockHD.VerMayor := CVERMAYOR;
  vBlockHD.VerMinor := CVERMINOR;
end;

procedure InitDataHD(var vDataHD: TKDataHeader);
begin
  FillChar(vDataHD, SizeOf(vDataHD), 0);
  vDataHD.DataID := CDATAID;
  {vDataHD.FieldName := '';}
  vDataHD.Reserved4a := Ord('/');
  vDataHD.Reserved4b := Ord('|');
end;

procedure TKBlockObject.SaveDataToStream(Stream: TStream);
var vBlockHD: TKBlockHeader;
  vDataHD: TKDataHeader;
  vData: TKFieldObject;
  vFieldName: string;
  i: Integer;
begin
  InitBlockHD(vBlockHD);
  vBlockHD.ObjCount := FFieldList.Count;
  vBlockHD.BlockFileId := FBlockFileId;
  Stream.Write(vBlockHD, SizeOf(vBlockHD));
  for i:=0 to FFieldList.Count-1 do begin
    vData := FFieldList[i];
    InitDataHD(vDataHD);
    vFieldName := vData.FFieldName;
    vDataHD.NameLength := Length(vFieldName);
    vDataHD.TagValue1 := vData.FTagValue1;
    vDataHD.TagValue2 := vData.FTagValue2;
    vDataHD.TagValue3 := vData.FTagValue3;
    vData.FillDataHeader(vDataHD);
    Stream.Write(vDataHD, SizeOf(vDataHD));
    if vDataHD.NameLength>0 then begin
      WriteString(Stream, vFieldName);
```

```
    end;
    if vDataHD.DataSize>0 then begin
      vData.WriteStream(Stream);
    end;
  end;
end;

procedure TKBlockObject.LoadDataFromStream(Stream: TStream);
var vBlockHD: TKBlockHeader;
  vDataHD: TKDataHeader;
  vData: TKFieldObject;
  vCount: Longint;
  i: Integer;
  vFieldName: string;
begin
  ClearFields;
  FillChar(vBlockHD, SizeOf(vBlockHD), 0);
  vCount := Stream.Read(vBlockHD, SizeOf(vBlockHD));
  if vCount=SizeOf(vBlockHD) then begin
    if (vBlockHD.BlockId = CBLOCKID) and (vBlockHD.DataId = CDATAID)
then begin
      FBlockFileId := vBlockHD.BlockFileId;
      for i:=0 to vBlockHD.ObjCount-1 do begin
        Stream.Read(vDataHD, SizeOf(vDataHD));
        if vDataHD.DataID = CDATAID then begin
          vFieldName := '';
          if vDataHD.NameLength>0 then begin
            ReadString(Stream, vFieldName, vDataHD.NameLength);
          end;
          vData := AddField(vDataHD.FieldCode, vFieldName);
          if vData<>nil then begin
            vData.FTagValue1 := vDataHD.TagValue1;
            vData.FTagValue2 := vDataHD.TagValue2;
            vData.FTagValue3 := vDataHD.TagValue3;
            vData.LoadDataHeader(vDataHD);
            if vDataHD.DataSize>0 then begin
              vData.ReadStream(Stream, vDataHD.DataSize);
            end;
          end else begin
            if vDataHD.DataSize>0 then begin
              Stream.Position := Stream.Position+vDataHD.DataSize;
            end;
          end;
        end else begin
          Raise Exception.Create(CInvalidBlockData);
        end;
      end;
    end else begin
      Raise Exception.Create(CInvalidBlockData);
    end;
  end else begin
    Raise Exception.Create(CInvalidBlockData);
  end;
end;
```

```
procedure TKBlockObject.SaveToStream(Stream: TStream);
begin
  RefreshBlockBinary;
  SaveDataToStream(Stream);
end;

procedure TKBlockObject.LoadFromStream(Stream: TStream);
begin
  LoadDataFromStream(Stream);
  RefreshBlockObject;
end;

function TKBlockObject.LoadFromFile(FileName: string): Integer;
var vMyStream: TMemoryStream;
begin
  vMyStream := TMemoryStream.Create;
  try
    vMyStream.LoadFromFile(FileName);
    vMyStream.Position := 0;
    LoadFromStream(vMyStream);
    result := BlockFileId;
  finally
    vMyStream.Free;
  end;
end;

procedure TKBlockObject.SaveToFile(FileName: string);
var vMyStream: TMemoryStream;
begin
  vMyStream := TMemoryStream.Create;
  try
    SaveToStream(vMyStream);
    vMyStream.Position := 0;
    vMyStream.SaveToFile(FileName);
  finally
    vMyStream.Free;
  end;
end;

function TKBlockObject.FieldByName(FieldName: TKFieldName):
TKFieldObject;
var i: Integer;
begin
  result := nil;
  i := 0 ;
  while i<= FFieldList.Count-1 do begin
    if CompareText(FieldName,
TKFieldObject(FFieldList[i]).FFieldName)=0 then begin
      result := FFieldList[i];
      i:=FFieldList.Count-1;
    end;
    inc(i);
  end;
end;
```

```
procedure TKBlockObject.RefreshBlockBinary;
var i: Integer;
  vField: TKFieldObject;
begin
  for i:=0 to FFieldList.Count-1 do begin
    vField := FFieldList[i];
    if vField is TKBlockField then begin
      TKBlockField(vField).FBlockData.RefreshBlockBinary;
      TKBlockField(vField).FBinaryData.Position := 0;
      TKBlockField(vField).FBinaryData.SetSize(0);

TKBlockField(vField).FBlockData.SaveDataToStream(TKBlockField(vField
).FBinaryData);
    end;
  end;
end;

procedure TKBlockObject.RefreshBlockObject;
var i: Integer;
  vField: TKFieldObject;
begin
  for i:=0 to FFieldList.Count-1 do begin
    vField := FFieldList[i];
    if vField is TKBlockField then begin
      TKBlockField(vField).FBinaryData.Position := 0;

TKBlockField(vField).FBlockData.LoadDataFromStream(TKBlockField(vFie
ld).FBinaryData);
      TKBlockField(vField).FBlockData.RefreshBlockObject;
    end;
  end;
end;

procedure TKBlockObject.MoveUp(Index: Integer);
var vField: TKFieldObject;
begin
  if Index>=1 then begin
    vField := FFieldList[Index-1];
    FFieldList[Index-1] := FFieldList[Index];
    FFieldList[Index] := vField;
  end;
end;

procedure TKBlockObject.MoveDown(Index: Integer);
var vField: TKFieldObject;
begin
  if Index<=FFieldList.Count-2 then begin
    vField := FFieldList[Index+1];
    FFieldList[Index+1] := FFieldList[Index];
    FFieldList[Index] := vField;
  end;
end;
```

```
function TKBlockObject.Modified: Boolean;
begin
  result := True;
end;

procedure TKBlockObject.SetTempInteger1(const Value: Integer);
begin
  FTempInteger1 := Value;
end;

procedure TKBlockObject.SetTempInteger2(const Value: Integer);
begin
  FTempInteger2 := Value;
end;

(* TKFieldObject *)

constructor TKFieldObject.Create(Owner: TKBlockObject);
begin
  inherited Create;
  if Owner=nil then begin
    raise Exception.Create('Invalid Owner');
  end else begin
    Owner.FFieldList.Add(Self);
    FOwner := Owner;
  end;
end;

destructor TKFieldObject.Destroy;
var vPos: Integer;
begin
  vPos := FOwner.FFieldList.IndexOf(Self);
  if vPos>=0 then begin
    FOwner.FFieldList.Delete(vPos);
  end;
  inherited;
end;

procedure TKFieldObject.FillDataHeader(var vDataHD: TKDataHeader);
begin
end;

procedure TKFieldObject.LoadDataHeader(vDataHD: TKDataHeader);
begin
end;

function TKFieldObject.GetDataSize: Longint;
begin
  result := 0;
end;

procedure TKFieldObject.WriteStream(Stream: TStream);
begin
end;
```

```
procedure TKFieldObject.ReadStream(Stream: TStream; Size: Longint);
begin
end;

procedure TKFieldObject.SetName(vName: String);
begin
  FFieldName := vName;
end;

(* TKIntegerField *)

constructor TKIntegerField.Create(Owner: TKBlockObject);
begin
  inherited;
  FIntegerData := 0;
end;

destructor TKIntegerField.Destroy;
begin
  inherited;
end;

procedure TKIntegerField.FillDataHeader(var vDataHD: TKDataHeader);
begin
  vDataHD.FieldCode := ClassId;
  vDataHD.IntegerData := FIntegerData;
  vDataHD.DataSize := 0;
end;

procedure TKIntegerField.LoadDataHeader(vDataHD: TKDataHeader);
begin
  FIntegerData := vDataHD.IntegerData;
end;

(* TKFloatField *)

constructor TKFloatField.Create(Owner: TKBlockObject);
begin
  inherited;
  FFloatData := 0;
end;

destructor TKFloatField.Destroy;
begin
  inherited;
end;

procedure TKFloatField.FillDataHeader(var vDataHD: TKDataHeader);
begin
  vDataHD.FieldCode := ClassId;
  vDataHD.FloatData := FFloatData;
  vDataHD.DataSize := 0;
end;
```

```
procedure TKFloatField.LoadDataHeader(vDataHD: TKDataHeader);
begin
  FFloatData := vDataHD.FloatData;
end;

(* TKStringField *)

constructor TKStringField.Create(Owner: TKBlockObject);
begin
  inherited;
  FStringData := '';
end;

destructor TKStringField.Destroy;
begin
  inherited;
end;

procedure TKStringField.FillDataHeader(var vDataHD: TKDataHeader);
begin
  vDataHD.FieldCode := ClassId;
  vDataHD.DataSize := GetDataSize;
end;

function TKStringField.GetDataSize: Longint;
begin
  result := Length(FStringData);
end;

procedure TKStringField.WriteStream(Stream: TStream);
begin
  if Length(FStringData)>0 then begin
    WriteString(Stream, FStringData);
  end;
end;

procedure TKStringField.ReadStream(Stream: TStream; Size: Longint);
begin
  ReadString(Stream, FStringData, Size);
end;

(* TKMemoField *)
(* nothing implemeneted *)

(* TKBinaryField *)

constructor TKBinaryField.Create(Owner: TKBlockObject);
begin
  inherited;
  FBinaryData := TMemoryStream.Create;
end;
```

```
destructor TKBinaryField.Destroy;
begin
  FBinaryData.Destroy;
  inherited;
end;

procedure TKBinaryField.FillDataHeader(var vDataHD: TKDataHeader);
begin
  vDataHD.FieldCode := ClassId;
  vDataHD.DataSize := GetDataSize;
end;

function TKBinaryField.GetDataSize: Longint;
begin
  result := FBinaryData.Size;
end;

procedure TKBinaryField.WriteStream(Stream: TStream);
begin
  if GetDataSize>0 then begin
    FBinaryData.Position := 0;
    Stream.CopyFrom(FBinaryData, GetDataSize);
  end;
end;

procedure TKBinaryField.ReadStream(Stream: TStream; Size: Longint);
begin
  FBinaryData.Position := 0;
  FBinaryData.SetSize(0);
  FBinaryData.CopyFrom(Stream, Size);
end;

procedure TKBinaryField.LoadFromStream(Stream: TStream);
begin
  FBinaryData.LoadFromStream(Stream);
end;

procedure TKBinaryField.SaveToStream(Stream: TStream);
begin
  FBinaryData.SaveToStream(Stream);
end;

procedure TKBinaryField.LoadFromFile(FileName: string);
begin
  FBinaryData.LoadFromFile(FileName);
end;

procedure TKBinaryField.SaveToFile(FileName: string);
begin
  FBinaryData.SaveToFile(FileName);
end;

(* TKGraphicField *)
(* Nothing Implemented *)

(* TKBlockField *)
```

```
constructor TKBlockField.Create(Owner: TKBlockObject);
begin
  inherited;
  FBlockData := TKBlockObject.Create;
  FBlockData.FOwnerField := Self;
end;

destructor TKBlockField.Destroy;
begin
  FBlockData.Destroy;
  inherited;
end;


end.
```

```
unit UDefault;

interface
uses Classes;
const
DefaultData: Array[0..1167] of char = (
  #075, #083, #066, #032, #061, #194, #002, #000, #007, #000, #000,
  #000, #000, #000, #000, #000, #001, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #061,
  #194, #002, #000, #000, #000, #000, #007, #146, #001, #000, #000,
  #000, #000, #000, #000, #015, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #047, #124, #068, #101, #102, #097, #117, #108, #116,
  #067, #114, #105, #116, #101, #114, #105, #097, #075, #083, #066,
  #032, #061, #194, #002, #000, #004, #000, #000, #000, #000, #000,
  #000, #000, #001, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #061, #194, #002, #000,
  #000, #000, #000, #004, #000, #000, #000, #000, #000, #000, #000,
  #000, #011, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #047,
  #124, #068, #101, #115, #099, #114, #105, #112, #116, #105, #111,
  #110, #061, #194, #002, #000, #000, #000, #000, #007, #032, #000,
  #000, #000, #000, #000, #000, #000, #008, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #047, #124, #067, #114, #105, #116, #101,
  #114, #105, #097, #075, #083, #066, #032, #061, #194, #002, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #001, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #061, #194, #002, #000, #000, #000, #000, #007, #032,
  #000, #000, #000, #000, #000, #000, #000, #006, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #047, #124, #077, #097, #116, #114,
  #105, #120, #075, #083, #066, #032, #061, #194, #002, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #001, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #061, #194, #002, #000, #000, #000, #000, #007, #032, #000,
  #000, #000, #000, #000, #000, #000, #009, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #047, #124, #073, #110, #100, #101, #112,
  #082, #101, #099, #116, #075, #083, #066, #032, #061, #194, #002,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #001, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #061, #194, #002, #000, #000, #000, #000, #007,
  #103, #000, #000, #000, #000, #000, #000, #000, #018, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
  #000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
```

```
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #047, #124, #068, #101, #102,
#097, #117, #108, #116, #065, #108, #116, #101, #114, #110, #097,
#116, #105, #118, #101, #075, #083, #066, #032, #061, #194, #002,
#000, #001, #000, #000, #000, #000, #000, #000, #000, #001, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #061, #194, #002, #000, #000, #000, #000, #004,
#000, #000, #000, #000, #000, #000, #000, #000, #011, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #047, #124, #068, #101, #115,
#099, #114, #105, #112, #116, #105, #111, #110, #061, #194, #002,
#000, #000, #000, #000, #007, #032, #000, #000, #000, #000, #000,
#000, #000, #006, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#047, #124, #077, #097, #116, #114, #105, #120, #075, #083, #066,
#032, #061, #194, #002, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #001, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #061, #194, #002, #000,
#000, #000, #000, #003, #000, #000, #000, #000, #000, #000, #000,
#000, #011, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #047,
#124, #068, #101, #115, #099, #114, #105, #112, #116, #105, #111,
#110, #061, #194, #002, #000, #000, #000, #000, #007, #032, #000,
#000, #000, #000, #000, #000, #000, #000, #008, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #047, #124, #067, #114, #105, #116, #101,
#114, #105, #097, #075, #083, #066, #032, #061, #194, #002, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #001, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #061, #194, #002, #000, #000, #000, #000, #007, #032,
#000, #000, #000, #000, #000, #000, #000, #011, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #047, #124, #065, #108, #116, #101,
#114, #110, #097, #116, #105, #118, #101, #075, #083, #066, #032,
#061, #194, #002, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #001, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #061, #194, #002, #000, #000,
#000, #000, #007, #032, #000, #000, #000, #000, #000, #000, #000,
#014, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #047, #124,
#076, #105, #110, #101, #097, #114, #080, #114, #111, #103, #068,
#097, #116, #097, #075, #083, #066, #032, #061, #194, #002, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #001, #000, #000,
#000, #000, #000, #000, #000, #000, #000, #000, #000, #000, #000,
```

```
  #000, #000);

var
  DefaultStream: TMemoryStream;

implementation

initialization
  DefaultStream := TMemoryStream.Create;
  DefaultStream.Write(DefaultData, SizeOf(DefaultData));

finalization
  DefaultStream.Free;

end.
```

```
unit UTreeDisplay;

interface

uses
  SysUtils, Windows, Messages, Classes, Graphics, UDataStru,
KFieldObjects,
  Math;

const
  C_MAXLEVEL = 20;
type
  TRTreeDisplayer = class
  private
    FCanvas: TCanvas;
    FDataStru: TRDataStru;
    FListLevel: TList;
    FXLeft: Integer;
    FDrawWidth: Integer;
    FDrawHeight :Integer;
    FArray: Array[0..C_MAXLEVEL] of TPoint;
    FMaxLevel: Integer;
    FLeftMargin: Integer;
    FTopMargin: Integer;
    FObjXDistance: Integer;
    FObjWidth: Integer;
    FObjHeight: Integer;
    FObjYDistance: Integer;
    FDrawValue: Boolean;
    FCX, FCY: Integer;
    FAlternativeColor: TColor;
    FCriteriaColor: TColor;
    FSelCriteriaColor: TColor;
    FSelAlternativeColor: TColor;
    FLineCriteriaColor: TColor;
    FLineAlternativeColor: TColor;
    FSelCriteria: TKBlockObject;
    FSelAlternative: TKBlockField;
    procedure SetDataStru(const Value: TRDataStru);
    procedure InsertToArray(vCrit: TKFieldObject; vLevel: Integer);
    procedure PreProcess(vCrit: TKBlockObject; vLevel: Integer);
    procedure PostProcess(vCrit: TKBlockObject; vLevel: Integer);
    procedure DoDrawLine(vCrit: TKBlockObject; vLevel: Integer);
    procedure DoDrawAlternative(vCrit: TKBlockObject; vLevel:
Integer);
    procedure DoDrawCriteria(vCrit: TKBlockObject; vLevel: Integer);
    procedure ClearListLevel;
    procedure SetLeftMargin(const Value: Integer);
    procedure SetTopMargin(const Value: Integer);
    procedure SetObjXDistance(const Value: Integer);
    procedure SetObjHeight(const Value: Integer);
    procedure SetObjWidth(const Value: Integer);
    procedure SetObjYDistance(const Value: Integer);
    procedure DrawObj(Canvas: TCanvas; vBlock: TKBlockObject);
```

```
virtual;
    procedure SetDrawValue(const Value: Boolean);
    procedure DrawLine(Canvas: TCanvas; vBlock: TKBlockObject);
virtual;
    procedure DrawAlternative;
    procedure DrawLineAlt(Canvas: TCanvas; vCrit: TKBLockObject);
    procedure SetAlternativeColor(const Value: TColor);
    procedure SetCriteriaColor(const Value: TColor);
    procedure SetSelAlternativeColor(const Value: TColor);
    procedure SetSelCriteriaColor(const Value: TColor);
    procedure SetLineAlternativeColor(const Value: TColor);
    procedure SetLineCriteriaColor(const Value: TColor);
    procedure SetSelAlternative(const Value: TKBlockField);
    procedure SetSelCriteria(const Value: TKBlockObject);
  public
    constructor Create;
    destructor Destroy; override;
    property SelCriteria: TKBlockObject read FSelCriteria write
SetSelCriteria;
    property SelAlternative: TKBlockField read FSelAlternative write
SetSelAlternative;
    property CriteriaColor: TColor read FCriteriaColor write
SetCriteriaColor;
    property AlternativeColor: TColor read FAlternativeColor write
SetAlternativeColor;
    property SelCriteriaColor: TColor read FSelCriteriaColor write
SetSelCriteriaColor;
    property SelAlternativeColor: TColor read FSelAlternativeColor
write SetSelAlternativeColor;
    property LineCriteriaColor: TColor read FLineCriteriaColor write
SetLineCriteriaColor;
    property LineAlternativeColor: TColor read FLineAlternativeColor
write SetLineAlternativeColor;
    property TopMargin: Integer read FTopMargin write SetTopMargin;
    property LeftMargin: Integer read FLeftMargin write
SetLeftMargin;
    property ObjXDistance: Integer read FObjXDistance write
SetObjXDistance;
    property ObjYDistance: Integer read FObjYDistance write
SetObjYDistance;
    property ObjWidth: Integer read FObjWidth write SetObjWidth;
    property ObjHeight: Integer read FObjHeight write SetObjHeight;
    property DrawWidth: Integer read FDrawWidth;
    property DrawHeight: Integer read FDrawHeight;
    property DrawValue: Boolean read FDrawValue write SetDrawValue;
    property Canvas: TCanvas read FCanvas write FCanvas;
    property DataStru: TRDataStru read FDataStru write SetDataStru;
    procedure SelectXY(X, Y: Integer);
    procedure Draw; virtual;
  end;

implementation

{ TRTreeDisplayer }
```

```
procedure TextDraw(Canvas: TCanvas; S: string; Style: Word; var
xRect: TRect);
var xchar: Array[0..255] of Char;
begin
  StrPCopy(xchar, S);
  DrawText(Canvas.Handle, xchar, -1, xRect, Style);
end;


procedure TRTreeDisplayer.ClearListLevel;
var i: Integer;
  vList: TList;
begin
  for i:=0 to FListLevel.Count-1 do begin
    vList := FListLevel[i];
    vList.Clear;
  end;
end;

constructor TRTreeDisplayer.Create;
var i: Integer;
  vList: TList;
begin
  FTopMargin := 20;
  FLeftMargin := 20;
  FObjWidth := 100;
  FObjXDistance := 5;
  FObjYDistance := 30;
  FObjHeight := 40;
  FCriteriaColor := clWindow;
  FAlternativeColor := clAqua;
  FSelCriteriaColor := clYellow;
  FSelAlternativeColor := clTeal;
  FLineCriteriaColor := clBlack;
  FLineAlternativeColor := clLime;
  FListLevel := TList.Create;
  for i:=0 to C_MAXLEVEL-1 do begin
    vList := TList.Create;
    FListLevel.Add(vList);
  end;
end;

destructor TRTreeDisplayer.Destroy;
var i: Integer;
  vList: TList;
begin
  for i:=0 to FListLevel.Count-1 do begin
    vList := FListLevel[i];
    vList.Free;
  end;
  FListLevel.Free;
  inherited;
end;
```

```
procedure TRTreeDisplayer.Draw;
begin
  Canvas.Font.Name := 'arial';
  Canvas.Font.Size := 8;
  FXLeft := 0;
  FMaxLevel := 0;
  FDataStru.TraceTree(nil, PreProcess, PostProcess, True);
  FDrawWidth := Max(FXLeft+FLeftmargin, 600);
  FDrawHeight :=
(FMaxLevel+4)*(FObjHeight+FObjYDistance)+FTopMargin;
  FXLeft := 0;
  DrawAlternative;
  FDataStru.TraceTree(nil, nil, DoDrawAlternative, True);
  FDataStru.TraceTree(nil, nil, DoDrawLine, True);
  FDataStru.TraceTree(nil, nil, DoDrawCriteria, True);
  DrawAlternative;
end;

procedure TRTreeDisplayer.InsertToArray(vCrit: TKFieldObject;
  vLevel: Integer);
var vList: TList;
begin
  if (vLevel>=0) and (vLevel<C_MAXLEVEL) then begin
    vList := FListLevel[vLevel];
    vList.Add(vCrit);
  end;
end;

procedure TRTreeDisplayer.PostProcess(vCrit: TKBlockObject;
  vLevel: Integer);
var vx, vy: Integer;
  vFIeld: TKFieldObject;
  i: Integer;
begin
  if (FCanvas=nil) or (FDataStru=nil) then exit;
  FMaxLevel := Max(FMaxLevel, vLevel);
  if vCrit.OwnerField.Owner.OwnerField<>nil then begin
    vFIeld := vCrit.OwnerField.Owner.OwnerField;
    {Memo1.Lines.Add('3-'+vField.FieldName+
      '-'+IntToStr(vCrit.FieldCount));}
  end else begin
    //Memo1.Lines.Add('3-Goal');
    vField := nil;
  end;
  if (vField<>nil) then begin
    if (vCrit.FieldCount=0) then begin
      vx := FXLeft+FLeftMargin;
      vy := vLevel*(FObjHeight+FObjYDistance)+FTopMargin;
      vCrit.TempInteger1 := vx;
      vCrit.TempInteger2 := vy;
      DrawObj(Canvas, vCrit);
      {Canvas.Rectangle(vx, vy, vx+FObjWidth, vy+FObjHeight);
      Canvas.TextOut(vx, vy, vField.FieldName);}
      FXLeft := FXLeft+FObjXDistance+FObjWidth;
      if vLevel>0 then begin
        FArray[vLevel-1].x := Min(FArray[vLevel-1].x, vx);
```

```
        FArray[vLevel-1].y := Max(FArray[vLevel-1].y, vx);
      end;
      //FMaxWidth := Max(FMaxWidth, );
    end else begin
      vx := ((FArray[vLevel].x+FArray[vLevel].y) div 2);
      vy := vLevel*(FObjHeight+FObjYDistance)+FTopMargin;
      vCrit.TempInteger1 := vx;
      vCrit.TempInteger2 := vy;
      DrawObj(Canvas, vCrit);
      {Canvas.Rectangle(vx, vy, vx+FObjWidth, vy+FObjHeight);
      Canvas.TextOut(vx, vy, vField.FieldName);}
      if vLevel>0 then begin
        FArray[vLevel-1].x := Min(FArray[vLevel-1].x, vx);
        FArray[vLevel-1].y := Max(FArray[vLevel-1].y, vx);
      end;
    end;
  end else begin
    vx := ((FArray[0].x+FArray[0].y) div 2){-FObjXDistance*2};
    vy := vLevel*(FObjHeight+FObjYDistance)+FTopMargin;
    DrawObj(Canvas, vCrit);
    vCrit.TempInteger1 := vx;
    vCrit.TempInteger2 := vy;
    vx := ((FArray[vLevel].x+FArray[vLevel].y) div 2);
    vy := vLevel*(FObjHeight+FObjYDistance)+FTopMargin;
    {Canvas.Rectangle(vx, vy, vx+FObjWidth, vy+FObjHeight);
    Canvas.TextOut(vx, vy, 'Goal');}
  end;
end;

procedure TRTreeDisplayer.PreProcess(vCrit: TKBlockObject;
  vLevel: Integer);
begin
  FArray[vLevel].x := 99999;
  FArray[vLevel].y := 0;
end;

procedure TRTreeDisplayer.SetDataStru(const Value: TRDataStru);
begin
  FDataStru := Value;
end;


procedure TRTreeDisplayer.SetLeftMargin(const Value: Integer);
begin
  FLeftMargin := Value;
end;

procedure TRTreeDisplayer.SetObjXDistance(const Value: Integer);
begin
  FObjXDistance := Value;
end;

procedure TRTreeDisplayer.SetObjHeight(const Value: Integer);
begin
  FObjHeight := Value;
end;
```

```
procedure TRTreeDisplayer.SetObjWidth(const Value: Integer);
begin
  FObjWidth := Value;
end;

procedure TRTreeDisplayer.SetTopMargin(const Value: Integer);
begin
  FTopMargin := Value;
end;

procedure TRTreeDisplayer.SetObjYDistance(const Value: Integer);
begin
  FObjYDistance := Value;
end;

procedure TRTreeDisplayer.DrawObj(Canvas: TCanvas; vBlock:
TKBlockObject);
var vTeks: string;
  vRect, xRect: TRect;
  vTopLevel: Boolean;
  vx, vy: Integer;
begin
  vTeks := '';
  vTopLevel := False;
  if vBLock = FSelCriteria then begin
    Canvas.Brush.Color := FSelCriteriaColor;
  end else begin
    Canvas.Brush.Color := FCriteriaColor;
  end;
  Canvas.Pen.Color := clBlack;
  try
    if vBlock.OwnerField.Owner.OwnerField<>nil then begin
      vTeks :=
TKBlockField(vBlock.OwnerField).Owner.OwnerField.FieldName;
    end else begin
      vTeks := 'Goal';
    end;
  except
    vTeks := 'Goal';
    vTopLevel := True;
  end;
  vx := vBlock.TempInteger1;
  vy := vBlock.TempInteger2;
  vRect := Rect(vx, vy, vx+FObjWidth, vy+FObjHeight);
  Canvas.Rectangle(vx, vy, vx+FObjWidth, vy+FObjHeight);
  if FDrawValue then begin
    TextDraw(Canvas, vTeks, DT_CENTER+DT_VCENTER+DT_SINGLELINE,
vRect);
  end else begin
    xRect := vRect;
    xRect.Bottom := (xRect.Bottom+xRect.Top) div 2;
    TextDraw(Canvas, vTeks, DT_CENTER+DT_VCENTER+DT_SINGLELINE,
xRect);
    Canvas.MoveTo(xRect.Left, xRect.Bottom);
    Canvas.LineTo(xRect.Right, xRect.Bottom);
```

```
    xRect := vRect;
    xRect.Top := (xRect.Bottom+xRect.Top) div 2;
    xRect.Right := (xRect.Right+xRect.Left) div 2;
    //vTeks := FormatFloat('G: '+'0.####', vBlock.TempReal1);
    vTeks := FormatFloat('G: '+'0.####', vBlock.TempReal3);
    TextDraw(Canvas, vTeks, DT_CENTER+DT_VCENTER+DT_SINGLELINE,
xRect);
    Canvas.MoveTo(xRect.Right, xRect.Top);
    Canvas.LineTo(xRect.Right, xRect.Bottom);
    xRect := vRect;
    xRect.Top := (xRect.Bottom+xRect.Top) div 2;
    xRect.Left := (xRect.Right+xRect.Left) div 2;
    //vTeks := FormatFloat('L: '+'0.####', vBlock.TempReal2);
    vTeks := FormatFloat('L: '+'0.####', vBlock.TempReal4);
    TextDraw(Canvas, vTeks, DT_CENTER+DT_VCENTER+DT_SINGLELINE,
xRect);
  end;
end;

procedure TRTreeDisplayer.SetDrawValue(const Value: Boolean);
begin
  FDrawValue := Value;
end;

procedure TRTreeDisplayer.DrawLine(Canvas: TCanvas; vBlock:
TKBlockObject);
var vx1, vy1, vx2, vy2: Integer;
begin
  try
    Canvas.Pen.Color := FLineCriteriaColor;
    vx1 := vBlock.TempInteger1;
    vx2 := vBlock.OwnerField.Owner.OwnerField.Owner.TempInteger1;
    vy1 := vBlock.TempInteger2;
    vy2 := vBlock.OwnerField.Owner.OwnerField.Owner.TempInteger2;
    //Canvas.Rectangle(vx1-5, vy1-5, vx1+5, vy1+5);
    Canvas.MoveTo(vx1+FObjWidth div 2, vy1);
    Canvas.LineTo(vx2+FObjWidth div 2, vy2+FObjHeight);
  except
  end;
end;

procedure TRTreeDisplayer.DoDrawLine(vCrit: TKBlockObject;
  vLevel: Integer);
begin
  DrawLine(Canvas, vCrit);
end;

procedure TRTreeDisplayer.DoDrawAlternative(vCrit: TKBlockObject;
  vLevel: Integer);
var vx, vy: Integer;
  vFIeld: TKFieldObject;
  i: Integer;
begin
  if (FCanvas=nil) or (FDataStru=nil) then exit;
  FMaxLevel := Max(FMaxLevel, vLevel);
  if vCrit.OwnerField.Owner.OwnerField<>nil then begin
```

```
    vFIeld := vCrit.OwnerField.Owner.OwnerField;
    {Memo1.Lines.Add('3-'+vField.FieldName+
      '-'+IntToStr(vCrit.FieldCount));}
  end else begin
    //Memo1.Lines.Add('3-Goal');
    vField := nil;
  end;
  if (vField<>nil) then begin
    if (vCrit.FieldCount=0) then begin
      DrawLineAlt(Canvas, vCrit);
      {Canvas.Rectangle(vx, vy, vx+FObjWidth, vy+FObjHeight);
      Canvas.TextOut(vx, vy, vField.FieldName);}
      //FMaxWidth := Max(FMaxWidth, );
    end else begin
    end;
  end else begin
  end;
end;

procedure TRTreeDisplayer.DrawAlternative;
var vLeft: Integer;
  vx, vy: Integer;
  vField: TKBLockField;
  i: Integer;
  vWidth: Integer;
  vPart: Integer;
  vRect: TRect;
  xRect: TRect;
begin
  Canvas.Pen.Color := clBlack;
  vLeft := (FDrawWidth-
(FDataStru.AltList.FieldCount)*(FObjWidth+FObjXDistance)) div 2;
  for i:=0 to FDataStru.AltList.FieldCount-1 do begin
    vField := TKBlockField(FDataStru.AltList[i]);
    vx := vLeft;
    vy :=
(FMaxLevel+1)*(FObjHeight+FObjYDistance)+FTopMargin+FObjHeight;
    vLeft := vLeft+FObjWidth+FObjXDistance;
    if FSelAlternative = vField then begin
      Canvas.Brush.COlor := FSelAlternativeColor;
    end else begin
      Canvas.Brush.COlor := FAlternativeColor;
    end;
    vField.BlockData.TempInteger1 := vx;
    vField.BlockData.TempInteger2 := vy;
    Canvas.Rectangle(vx, vy, vx+FObjWidth, vy+FObjHeight);
    vRect := Rect(vx, vy, vx+FObjWidth, vy+FObjHeight);
    xRect := vRect;
    if DrawValue then begin
      TextDraw(Canvas, vField.FieldName,
DT_CENTER+DT_VCENTER+DT_SINGLELINE, xRect);
    end else begin
      xRect := vRect;
      xRect.Bottom := (vRect.Top+vRect.Bottom) div 2;
      Canvas.MoveTo(xRect.Left, xRect.Bottom);
      Canvas.LineTo(xRect.Right, xRect.Bottom);
```

```
      TextDraw(Canvas, vField.FieldName,
DT_CENTER+DT_VCENTER+DT_SINGLELINE, xRect);
      xRect := vRect;
      xRect.Top := (vRect.Top+vRect.Bottom) div 2;
      TextDraw(Canvas, FormatFloat('0.####',
vField.BlockData.TempReal1), DT_CENTER+DT_VCENTER+DT_SINGLELINE,
xRect);
    end;


    //Canvas.Rectangle(10, 10, 100, 100);
  end;
{  if FDataStru.AltList.FieldCount >0 then begin
    vWidth := FDrawWidth-FLeftMargin*2;
    vPart := vWidth div FDataStru.AltList.FieldCount;
    for i:=0 to FDataStru.AltList.FieldCount-1 do begin
      vField := TKBlockField(FDataStru.AltList[i]);
      vx := (i*vPart)+(vPart-FObjHeight) div 2;
      vy := (FMaxLevel+1)*(FObjHeight+FObjYDistance)+FTopMargin;
      vField.BlockData.TempInteger1 := vx;
      vField.BlockData.TempInteger2 := vy;
      Canvas.Rectangle(vx, vy, vx+FObjWidth, vy+FObjHeight);
      //Canvas.Rectangle(10, 10, 100, 100);
    end;
  end;}

end;

procedure TRTreeDisplayer.DrawLineAlt(Canvas: TCanvas;
  vCrit: TKBLockObject);
var i: Integer;
  vField: TKFieldObject;
  vBlock: TKBlockObject;
  vx1, vy1, vx2, vy2: Integer;
begin
  Canvas.Pen.Color := FLineAlternativeColor;
  vx1 := vCrit.TempInteger1+FObjWidth div 2+FLeftMargin;
  vy1 := vCrit.TempInteger2+FObjHeight;
  for i:=0 to FDataStru.AltList.FieldCount-1 do begin
    vField := FDataStru.AltList[i];
    vx2 := TKBlockField(vField).BlockData.TempInteger1+FObjWidth div
2;
    vy2 := TKBlockField(vField).BlockData.TempInteger2;
    Canvas.MoveTo(vx1, vy1);
    Canvas.LineTo(vx2, vy2);
  end;
end;

procedure TRTreeDisplayer.DoDrawCriteria(vCrit: TKBlockObject;
  vLevel: Integer);
begin
  DrawObj(Canvas, vCrit);
end;
```

```
procedure TRTreeDisplayer.SetAlternativeColor(const Value: TColor);
begin
  FAlternativeColor := Value;
end;

procedure TRTreeDisplayer.SetCriteriaColor(const Value: TColor);
begin
  FCriteriaColor := Value;
end;

procedure TRTreeDisplayer.SetSelAlternativeColor(const Value:
TColor);
begin
  FSelAlternativeColor := Value;
end;

procedure TRTreeDisplayer.SetSelCriteriaColor(const Value: TColor);
begin
  FSelCriteriaColor := Value;
end;

procedure TRTreeDisplayer.SetLineAlternativeColor(const Value:
TColor);
begin
  FLineAlternativeColor := Value;
end;

procedure TRTreeDisplayer.SetLineCriteriaColor(const Value: TColor);
begin
  FLineCriteriaColor := Value;
end;

procedure TRTreeDisplayer.SetSelAlternative(const Value:
TKBlockField);
begin
  FSelAlternative := Value;
end;

procedure TRTreeDisplayer.SetSelCriteria(const Value:
TKBlockObject);
begin
  FSelCriteria := Value;
end;

procedure TRTreeDisplayer.SelectXY(X, Y: Integer);
var i: Integer;
  vBlock: TKBLockObject;
  vField: TKFieldObject;
  vx, vy: Integer;
  vRect: TRect;
begin
  for i:=0 to FDataStru.CritList.Count-1 do begin
    vBlock := FDataStru.CritList[i];
    vx := vBlock.TempInteger1;
    vy := vBlock.TempInteger2;
    vRect := Rect(vx, vy, vx+FObjWidth, vy+FObjHeight);
```

```
    if PtInRect(vRect, Point(x, y)) then begin
      FSelCriteria:= vBlock;
      Draw;
      Exit;
    end;
  end;
  for i:=0 to FDataStru.AltList.FieldCount-1 do begin
    vBlock := TKBlockField(FDataStru.AltList[i]).BlockData;
    vx := vBlock.TempInteger1;
    vy := vBlock.TempInteger2;
    vRect := Rect(vx, vy, vx+FObjWidth, vy+FObjHeight);
    if PtInRect(vRect, Point(x, y)) then begin
      FSelAlternative := TKBlockField(FDataStru.AltList[i]);
      Draw;
      Exit;
    end;
  end;
  FSelCriteria := nil;
  FSelAlternative := nil;
  Draw;
end;

end.
```

```
unit Fdlgdat;

interface

uses WinTypes, WinProcs, Classes, Graphics, Forms, Controls,
Buttons,
  StdCtrls, ExtCtrls, Dialogs;

type
  TDialogData = class(TForm)
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
    Bevel1: TBevel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Memo1: TMemo;
    procedure FormCloseQuery(Sender: TObject; var CanClose:
Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
    DataCheck: function(vtest: string): Boolean of object;
  end;

var
  DialogData: TDialogData;

implementation

{$R *.DFM}

procedure TDialogData.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
begin
  if ModalResult = mrOK then begin
    if Pos(',', Edit1.Text)>0 then begin
      ShowMessage('"," tidak diijinkan pada Nama');
      CanClose := False;
      exit;
    end;
    if Edit1.Text = '' then begin
      ShowMessage('Data Id tidak boleh Kosong');
      CanClose := False;
    end else begin
      if Assigned(DataCheck) then begin
        if DataCheck(Edit1.Text) then begin
          ShowMessage('Data Id sudah ada');
          CanClose := False;
        end;
      end;
    end;
  end;
```

```
end;
end.
```

```
unit Fpilihan;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, StdCtrls, Buttons;

type
  TFormPilihan = class(TForm)
    Label1: TLabel;
    RadioInteger: TRadioButton;
    RadioReal: TRadioButton;
    BitBtn1: TBitBtn;
    RadioButton1: TRadioButton;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormPilihan: TFormPilihan;

implementation

{$R *.DFM}

end.
```

```
unit Fquest2;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, ExtCtrls, StdCtrls, KFieldObjects, UDataStru,
Grids,
  GlobalUnit, ComCtrls;

type
  TRdGroup = class(TPanel)
  private
    FNilai: Integer;
    FTanda: Integer;
    FOnChange: TNotifyEvent;
  protected
    procedure CreateButton(Mode: Integer);
    procedure ChangeData(Sender: TObject);
  public
    property OnChange: TNotifyEvent read FOnChange write FOnChange;
    procedure SetNilai(vTanda, vNilai: Integer);
  end;

  TEditGroup = class(TPanel)
  private
    FNilai: Extended;
    FTanda: Integer;
    FOnChange: TNotifyEvent;
  protected
    procedure CreateButton(Mode: Integer);
    procedure ChangeData(Sender: TObject);
    procedure KeyIsPressed(Sender: TObject; var Key: Char);
    procedure ExitData(Sender: TObject);
  public
    property OnChange: TNotifyEvent read FOnChange write FOnChange;
    procedure SetNilai(vTanda: Integer; vNilai: Extended);
  end;

  TTrackGroup = class(TPanel)
  private
    FNilai: Extended;
    FTanda: Integer;
    FOnChange: TNotifyEvent;
    FLbl1, FLbl2: TLabel;
    FShape1, FShape2: TLabel;
  protected
    procedure CreateButton(Mode: Integer);
    procedure ChangeData(Sender: TObject);
  public
    property OnChange: TNotifyEvent read FOnChange write FOnChange;
    procedure SetNilai(vTanda: Integer; vNilai: Extended);
  end;
```

```
  TFormQuest2 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    ScrollBox1: TScrollBox;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel5: TPanel;
    Panel6: TPanel;
    Panel7: TPanel;
    Panel8: TPanel;
    Panel10: TPanel;
    ComboBox1: TComboBox;
    ScrollBar1: TScrollBar;
    DrawGrid1: TDrawGrid;
    Splitter1: TSplitter;
    Button5: TButton;
    BtnPrev: TButton;
    BtnNext: TButton;
    Button4: TButton;
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    StatusBar1: TStatusBar;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure BtnPrevClick(Sender: TObject);
    procedure BtnNextClick(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
    procedure DrawGrid1DrawCell(Sender: TObject; ACol, ARow:
Integer;
      Rect: TRect; State: TGridDrawState);
    procedure Button5Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
    FListElement: TList;
    FListObj: TList;
    FCurrentCrit: TKBlockObject;
    FDataStru: TRDataStru;
    FStatus: Integer;
    FItemList: TStringList;
    FCritList: TStringList;
    FTemporerArray: TArrayCriteria;
    FDataUpdated: Boolean;
    FDontChange: Boolean;
    FIndepChanged: Boolean;
    procedure ShowUpdateInfo;
    procedure SetCurrentCrit(const Value: TKBlockObject);
    procedure SetDataStru(const Value: TRDataStru);
    procedure SetStatus(const Value: Integer);
    procedure SetDataUpdated(const Value: Boolean);
    procedure SetIndepChanged(const Value: Boolean);
```

```
  public
    { Public declarations }
    Refreshing: TNotifyEvent;
    StatusEntry: Integer;
    DataChange: TNotifyEvent;
    property DataUpdated: Boolean read FDataUpdated write
SetDataUpdated;
    property IndepChanged: Boolean read FIndepChanged write
SetIndepChanged;
    property Status: Integer read FStatus write SetStatus;
    procedure BuildEntry;
    property DataStru: TRDataStru read FDataStru write SetDataStru;
    property CurrentCrit: TKBlockObject read FCurrentCrit write
SetCurrentCrit;
    procedure GetData(indeks: Integer; var Tanda: Integer;
      var Nilai: Extended);
    procedure SetData(indeks: Integer; Tanda: Integer; Nilai:
Extended);
    property ListElement: TList read FListElement;
    procedure CreateCompareObj(Jenis: Integer; vCurrent: string;
      vStrList: TStrings);
    procedure CalculatingData;
  end;

var
  FormQuest2: TFormQuest2;

implementation

uses Fvalid, Fguide;

{$R *.DFM}

(* TRdGroup *)

procedure TRdGroup.ChangeData(Sender: TObject);
var vIndeks: Integer;
begin
  if Sender is TRadioButton then begin
    vIndeks := TRadioButton(Sender).Tag;
    if vIndeks<8 then begin
      FNilai := 9-vIndeks;
      FTanda := 1;
    end else begin
      FNilai := vIndeks-7;
      FTanda := 0;
    end;
  end;
  if Assigned(FOnChange) then begin
    FOnChange(Self);
  end;
end;
```

```
procedure TRdGroup.SetNilai(vTanda, vNilai: Integer);
var vComponent: TComponent;
  vNomor: Integer;
begin
  FNilai := vNilai;
  FTanda := vTanda;
  if vTanda = 0 then begin
    vNomor := vNilai+9-2;
  end else begin
    vNomor := 9-vNilai;
  end;
  vComponent := Components[vNomor];
  if vComponent is TRadioButton then begin
    TRadioButton(vComponent).Checked := True;
  end;
end;

procedure TRdGroup.CreateButton(Mode: Integer);
var vRBtn: TRadioButton;
  i: Integer;
  vValue: Integer;
  cLebarRadio: Integer;
  cKiri, cAtas: Integer;
  vLbl: TLabel;
begin
  cLebarRadio := 18;
  cKiri := (Width - cLebarRadio*17) div 2;
  cAtas := 5;
  for i:=0 to 16 do begin
    if i<8 then begin
      vValue := 9-i;
    end else begin
      vValue := i-7;
    end;
    if Mode=0 then begin
      vRBtn := TRadioButton.Create(Self);
      vRBtn.Parent := Self;
      vRBtn.Left := i*cLebarRadio+cKiri;
      vRBtn.Width := cLebarRadio-2;
      vRBtn.Top := cAtas;
      vRBtn.OnClick := ChangeData;
      vRBtn.Caption := ''{IntToStr(vValue)};
      vRBtn.Tag := i;
    end else begin
      vLbl := TLabel.Create(Self);
      vLbl.AutoSize := False;
      vLbl.Parent := Self;
      vLbl.Left := i*cLebarRadio+cKiri;
      vLbl.Width := cLebarRadio-2;
      vLbl.Alignment := taCenter;
      vLbl.Top := cAtas;
      vLbl.Caption := IntToStr(vValue);
    end;
  end;
end;
```

```
(* TEditGroup *)

procedure TEditGroup.KeyIsPressed(Sender: TObject; var Key: Char);
begin
  if Pos(Key, '0123456789.-'#8)<=0 then begin
    Key := #0;
  end;
end;

procedure TEditGroup.ExitData(Sender: TObject);
begin
  if Sender is TEdit then begin
    if (FNilai<1)then begin
      TEdit(Sender).Text := '1';
      ChangeData(Sender);
    end else if (FNilai>9) then begin
      TEdit(Sender).Text := '9';
      ChangeData(Sender);
    end;
  end;
end;

procedure TEditGroup.ChangeData(Sender: TObject);
var vComponent0, vComponent1: TComponent;
  vNilai: Extended;
  vKode: Integer;
  vTeks: string;
  vError: Integer;
begin
  vComponent0 := Components[0];
  vComponent1 := Components[1];
  vError := 0;
  vKode := 0;
  if (vComponent0 is TEdit) and (vComponent1 is TComboBox) then
begin
    FTanda := TComboBox(vComponent1).ItemIndex;
    if FTanda<0 then begin
      FTanda := 0;
    end;
    vTeks := TEdit(vComponent0).Text;
    if vTeks = '' then begin
      FNilai := 0;
    end else begin
      Val(vTeks, vNilai, vKode);
      FNilai := vNilai;
      {if vKode <> 0 then begin
        vError := 1;
      end else if (vNilai<1) or (vNilai>9) then begin
        vError := 2;
      end;}
    end;
    if Assigned(FOnChange) then begin
      FOnChange(Self);
    end;
  end;
end;
```

```
procedure TEditGroup.SetNilai(vTanda: Integer; vNilai: Extended);
var vComponent: TComponent;
  vNomor: Integer;
begin
  FNilai := vNilai;
  FTanda := vTanda;
  vComponent := Components[0];
  if vComponent is TEdit then begin
    TEdit(vComponent).Text := FormatFloat('#0.##', vNilai);
  end;
  vComponent := Components[1];
  if vComponent is TComboBox then begin
    TComboBox(vComponent).ItemIndex := vTanda;
  end;

end;

procedure TEditGroup.CreateButton(Mode: Integer);
var vEdit: TEdit;
  vCombo: TComboBox;
  cLebar, cKiri: Integer;
  cTinggi: Integer;
begin
  BevelOuter := bvNone;
  cKiri := 5;
  cTinggi := Width-2*cKiri;
  cLebar := (Width - cKiri*3) div 2;
  vEdit := TEdit.Create(Self);
  vEdit.Parent := Self;
  vEdit.SetBounds(cKiri, cKiri, cLebar, cTinggi);
  vEdit.OnChange := ChangeData;
  vEdit.OnKeyPress := KeyIsPressed;
  vEdit.OnExit := ExitData;
  vCombo := TComboBox.Create(Self);
  vCombo.Parent := Self;
  vCombo.SetBounds(2*cKiri+cLebar, cKiri, cLebar, cTinggi);
  vCombo.Style := csDropDownList;
  vCombo.Items.Add('<');
  vCombo.Items.Add('>');
  vCombo.OnChange := ChangeData;
  vEdit.Height := vCombo.Height;
end;

(* TTrackGroup *)

procedure TTrackGroup.ChangeData(Sender: TObject);
var
  vPosition: Integer;
  vNilai1, vNilai2: Extended;
begin
  if Sender is TScrollBar then begin
    vPosition := TScrollBar(Sender).Position;
    if vPosition<800 then begin
      FNilai := 9-(vPosition/100);
      FTanda := 1;
```

```
    end else begin
      FNilai := (vPosition/100)-7;
      FTanda := 0;
    end;
  end;
  if FNilai = 0 then FNilai := 1;
  if FNilai<1 then begin
    FNilai := 1/FNilai;
    if FTanda = 0 then FTanda := 1 else FTanda := 0;
  end;
  if Assigned(FOnChange) then begin
    FOnChange(Self);
  end;
  if FTanda = 0 then begin
    vNilai1 := 1;
    vNilai2 := FNilai;
  end else begin
    vNilai1 := FNilai;
    vNilai2 := 1;
  end;
  FShape1.Width := Round(vNilai1*20);
  FShape2.Width := Round(vNilai2*20);
  FShape1.Caption := FormatFloat('#0.####', vNilai1);
  FShape2.Caption := FormatFloat('#0.####', vNilai2);
end;

procedure TTrackGroup.SetNilai(vTanda: Integer; vNilai: Extended);
var vComponent: TComponent;
  vNomor: Integer;
  vPosition: Integer;
  vNilai1: Extended;
  vNilai2: Extended;
begin
  if vNilai = 0 then vNilai := 1;
  if vNilai<1 then begin
    vNilai := 1/vNilai;
    if vTanda = 0 then vTanda := 1 else vTanda := 0;
  end;
  FNilai := vNilai;
  FTanda := vTanda;
  if vTanda = 0 then begin
    vPosition := Trunc(vNilai*100)+800-100;
    vNilai1 := 1;
    vNilai2 := vNilai;
  end else begin
    vPosition := 800-Trunc(FNilai*100)+100;
    vNilai1 := vNilai;
    vNilai2 := 1;
  end;
  vComponent := Components[0];
  TScrollBar(vComponent).Position := vPosition;
  FShape1.Width := Round(vNilai1*20);
  FShape2.Width := Round(vNilai2*20);
  FShape1.Caption := FormatFloat('#0.####', vNilai1);
  FShape2.Caption := FormatFloat('#0.####', vNilai2);
end;
```

```
procedure TTrackGroup.CreateButton(Mode: Integer);
var vRBtn: TRadioButton;
  i: Integer;
  vValue: Integer;
  cLebarRadio: Integer;
  cKiri, cAtas: Integer;
  vLbl: TLabel;
  vJmlLebar: Integer;
  vTrack: TScrollBar;
  vTop: Integer;
  vShape: TShape;
begin
  cLebarRadio := 17;
  cAtas := 5;
  if Mode = 0 then begin
    cKiri := ((Width - (cLebarRadio)*17) div 2)-17;
    vTrack := TScrollBar.Create(Self);
    vTrack.Left := cKiri;
    vTrack.Top := cAtas;
    vTrack.Width := ((cLebarRadio)*17)+34;
    vTrack.Max := 1600;
    vTrack.Parent := Self;
    vTrack.OnChange := ChangeData;
    FLbl1 := TLabel.Create(Self);
    FLbl1.Parent := Self;
    FLbl1.AutoSize := False;
    FLbl1.Transparent := False;
    FLbl1.Top := vTrack.Top+vTrack.Height+5;
    FLbl1.Left := 5;
    FLbl1.Height := 18;
    FLbl1.Width := 100;
    FLbl1.Color := clYellow;
    FShape1 := TLabel.Create(Self);
    FShape1.Parent := Self;
    FShape1.Color := clLime;
    FShape1.AutoSize := False;
    FShape1.Top := FLbl1.Top;
    FShape1.Height := FLbl1.Height;
    FShape1.Left := FLbl1.Width+10;
    {FShape1.Alignment := taCenter;}
    vTop := FLbl1.Top + FLbl1.Height;
    FLbl2 := TLabel.Create(Self);
    FLbl2.Parent := Self;
    FLbl2.Left := 5;
    FLbl2.AutoSize := False;
    FLbl2.Transparent := False;
    FLbl2.Top := vTop+5;
    FLbl2.Height := 18;
    FLbl2.Width := 100;
    FLbl2.Color := clYellow;
    FShape2 := TLabel.Create(Self);
    FShape2.Parent := Self;
    FShape2.AutoSize := False;
    FShape2.Color := clLime;
    FShape2.Top := FLbl2.Top;
```

```
    FShape2.Height := FLbl2.Height;
    FShape2.Left := FLbl2.Width+10;
    {FShape2.Alignment := taCenter;}
  end else begin
    cKiri := ((Width - (cLebarRadio)*17) div 2);
    for i:=0 to 16 do begin
      if i<8 then begin
        vValue := 9-i;
      end else begin
        vValue := i-7;
      end;
      vLbl := TLabel.Create(Self);
      vLbl.AutoSize := False;
      vLbl.Parent := Self;
      vLbl.Left := i*cLebarRadio+cKiri;
      vLbl.Width := cLebarRadio-2;
      vLbl.Alignment := taCenter;
      vLbl.Top := cAtas;
      vLbl.Caption := IntToStr(vValue);
    end;
  end;
end;


(* Form *)
procedure TFormQuest2.FormCreate(Sender: TObject);
begin
  FListObj := TList.Create;
  FListElement := TList.Create;
  StatusEntry := -1;
  Label1.Caption := '';
  FItemList := TStringList.Create;
  FCritList := TStringLIst.Create;

  {Button1Click(nil);}
end;

procedure TFormQuest2.FormDestroy(Sender: TObject);
begin
  FListObj.Free;
  FListElement.Free;
  FItemList.Free;
  FCritList.Free;
end;

function StringBagiDua(Data, Sparator: string; var vHasil1, vHasil2:
string): Boolean;
var
  vPos: Integer;
begin
  vPos := Pos(Sparator, Data);
  if vPos>0 then begin
    vHasil1 := Copy(Data, 1, vPos-1);
    vHasil2 := Copy(Data, vPos+Length(Sparator), Length(Data));
    result := True;
  end else begin
```

```
    vHasil1 := Data;
    vHasil2 := Data;
    result := False;
  end;
end;


procedure TFormQuest2.CreateCompareObj(Jenis: Integer; vCurrent:
string;
  vStrList: TStrings);
var vComponent: TComponent;
  vi, i: Integer;
  vTop: Integer;
  vDelta: Integer;
  vNilai: Extended;
  vTanda: Integer;
  caption1, caption2: string;
begin
  ScrollBox1.Visible := False;
  for i:=ScrollBox1.ComponentCount-1 downto 0 do begin
    vComponent := ScrollBox1.Components[i];
    vComponent.Free;
  end;
  if Jenis = 0 then begin
    vDelta := Panel3.Top-Panel5.Top;
    vComponent := TRdGroup.Create(ScrollBox1);
    //TControl(vComponent).Visible := False;
    TControl(vComponent).Parent := ScrollBox1;
    TControl(vComponent).SetBounds(Panel6.Left, Panel6.Top,
Panel6.Width,
      Panel6.Height);
    TRdGroup(vComponent).CreateButton(1);
    vi := 0;
    for i:=0 to vStrList.Count-1 do begin
      StringBagiDua(vStrList[i], ',', caption1, caption2);
      if CompareText(vStrList[i], vCurrent)<>0 then begin
        GetData(i, vTanda, vNilai);
        vTop := vi*Panel3.Height + Panel3.Top;
        {Left Panel}
        vComponent := TPanel.Create(ScrollBox1);
        //TControl(vComponent).Visible := False;
        TControl(vComponent).Parent := ScrollBox1;
        TControl(vComponent).SetBounds(Panel3.Left, vTop,
Panel3.Width, Panel3.Height);
        TPanel(vComponent).Caption := caption1;
        {Radio Buttons}
        vComponent := TRdGroup.Create(ScrollBox1);
        //TControl(vComponent).Visible := False;
        TControl(vComponent).Parent := ScrollBox1;
        TControl(vComponent).SetBounds(Panel5.Left, vTop-vDelta,
Panel5.Width,
          Panel5.Height);
        TRdGroup(vComponent).CreateButton(0);
        TRdGroup(vComponent).SetNilai(vTanda, Round(vNilai));
        TRdGroup(vComponent).OnChange := Edit1Change;
        vComponent.Tag := i;
```

```
        Edit1Change(vComponent);
        {TRadioGroup(vComponent).Columns := RadioGroup1.Columns;
        TRadioGroup(vComponent).Items.Assign(RadioGroup1.Items);}
        {Right Panel}
        vComponent := TPanel.Create(ScrollBox1);
        //TControl(vComponent).Visible := False;
        TControl(vComponent).Parent := ScrollBox1;
        TControl(vComponent).SetBounds(Panel4.Left, vTop,
Panel4.Width, Panel4.Height);
        TPanel(vComponent).Caption := caption2;
        inc(vi);
      end;
    end;
    if vStrList.Count>=9 then begin
      vDelta := (Panel5.Top)-(Panel6.Top+Panel6.Height);
      vTop := (vStrList.Count)*Panel3.Height + Panel3.Top;
      vComponent := TRdGroup.Create(ScrollBox1);
      //TControl(vComponent).Visible := False;
      TControl(vComponent).Parent := ScrollBox1;
      TControl(vComponent).SetBounds(Panel6.Left, vTop+vDelta,
Panel6.Width,
        Panel6.Height);
      TRdGroup(vComponent).CreateButton(1);
    end;
  end else if Jenis = 1 then begin
    vDelta := (Panel6.Height - Label1.Height);
    vComponent := TLabel.Create(ScrollBox1);
    //TControl(vComponent).Visible := False;
    TControl(vComponent).Parent := ScrollBox1;
    TLabel(vComponent).AutoSize := False;
    TLabel(vComponent).Alignment := taCenter;
    TControl(vComponent).SetBounds(Panel8.Left, Panel6.Top+vDelta,
      Panel8.Width div 2, Panel8.Height);
    TLabel(vComponent).Caption := 'Nilai';
    vComponent := TLabel.Create(ScrollBox1);
    //TControl(vComponent).Visible := False;
    TControl(vComponent).Parent := ScrollBox1;
    TLabel(vComponent).AutoSize := False;
    TLabel(vComponent).Alignment := taCenter;
    TControl(vComponent).SetBounds(Panel8.Left+Panel8.Width div 2,
      Panel6.Top+vDelta, Panel8.Width div 2, Panel8.Height);
    {TControl(vComponent).SetBounds(Panel9.Left, Panel6.Top+vDelta,
      Panel9.Width, Panel9.Height);}
    TLabel(vComponent).Caption := 'Arah';
    vDelta := (Panel8.Height - ComboBox1.Height) div 2;
    vi := 0;
    for i:=0 to vStrList.Count-1 do begin
      if CompareText(vStrList[i], vCurrent)<>0 then begin
        StringBagiDua(vStrList[i], ',', caption1, caption2);
        GetData(i, vTanda, vNilai);
        vTop := vi*Panel3.Height + Panel3.Top;
        {Left Panel}
        vComponent := TPanel.Create(ScrollBox1);
        //TControl(vComponent).Visible := False;
        TControl(vComponent).Parent := ScrollBox1;
```

```
        TControl(vComponent).SetBounds(Panel7.Left, vTop,
Panel7.Width, Panel7.Height);
        TPanel(vComponent).Caption := caption1;
        {Edit Panel}
        vComponent := TEditGroup.Create(ScrollBox1);
        //TControl(vComponent).Visible := False;
        TControl(vComponent).Parent := ScrollBox1;
        TControl(vComponent).SetBounds(Panel8.Left, vTop,
Panel8.Width,
          Panel8.Height);
        TEditGroup(vComponent).CreateButton(0);
        TEditGroup(vComponent).SetNilai(vTanda, vNilai);
        TEditGroup(vComponent).OnChange := Edit1Change;
        vComponent.Tag := i;
        {Edit Box}
        {vComponent := TEdit.Create(ScrollBox1);
        TControl(vComponent).Parent := ScrollBox1;
        TControl(vComponent).SetBounds(Panel8.Left, vTop+vDelta,
Panel8.Width, ComboBox1.Height);
        TEdit(vComponent).Text := FloatToStr(vNilai);
        vComponent := TComboBox.Create(ScrollBox1);
        TControl(vComponent).Parent := ScrollBox1;
        TControl(vComponent).SetBounds(Panel9.Left, vTop+vDelta,
Panel9.Width, ComboBox1.Height);
        TComboBox(vComponent).Style := csDropDownList;
        TComboBox(vComponent).Items.Assign(ComboBox1.Items);
        TComboBox(vComponent).ItemIndex := vTanda;}
        {Right Panel}
        vComponent := TPanel.Create(ScrollBox1);
        //TControl(vComponent).Visible := False;
        TControl(vComponent).Parent := ScrollBox1;
        TControl(vComponent).SetBounds(Panel10.Left, vTop,
Panel10.Width, Panel10.Height);
        TPanel(vComponent).Caption := caption2;
        inc(vi);
      end;
    end;
  end else begin
    vDelta := Panel3.Top-Panel5.Top;
    vComponent := TTrackGroup.Create(ScrollBox1);
    //TControl(vComponent).Visible := False;
    TControl(vComponent).Parent := ScrollBox1;
    TControl(vComponent).SetBounds(Panel6.Left, Panel6.Top,
Panel6.Width,
      Panel6.Height);
    TTrackGroup(vComponent).CreateButton(1);
    vi := 0;
    for i:=0 to vStrList.Count-1 do begin
      if CompareText(vStrList[i], vCurrent)<>0 then begin
        StringBagiDua(vStrList[i], ',', caption1, caption2);
        GetData(i, vTanda, vNilai);
        vTop := Trunc(vi*Panel3.Height*2.5) + Panel3.Top;
        {Left Panel}
        vComponent := TPanel.Create(ScrollBox1);
        //TControl(vComponent).Visible := False;
        TControl(vComponent).Parent := ScrollBox1;
```

```
        TControl(vComponent).SetBounds(Panel3.Left, vTop,
Panel3.Width, Panel3.Height);
        TPanel(vComponent).Caption := caption1;
        {Radio Buttons}
        vComponent := TTrackGroup.Create(ScrollBox1);
        //TControl(vComponent).Visible := False;
        TControl(vComponent).Parent := ScrollBox1;
        TControl(vComponent).SetBounds(Panel5.Left, vTop-vDelta,
Panel5.Width,
          Trunc(Panel5.Height*2.4));
        TTrackGroup(vComponent).CreateButton(0);
        TTrackGroup(vComponent).SetNilai(vTanda, vNilai);
        TTrackGroup(vComponent).OnChange := Edit1Change;
        TTrackGroup(vComponent).FLbl1.Caption := caption1;
        TTrackGroup(vComponent).FLbl2.Caption := caption2;
        vComponent.Tag := i;
        Edit1Change(vComponent);
        {TRadioGroup(vComponent).Columns := RadioGroup1.Columns;
        TRadioGroup(vComponent).Items.Assign(RadioGroup1.Items);}
        {Right Panel}
        vComponent := TPanel.Create(ScrollBox1);
        //TControl(vComponent).Visible := False;
        TControl(vComponent).Parent := ScrollBox1;
        TControl(vComponent).SetBounds(Panel4.Left, vTop,
Panel4.Width, Panel4.Height);
        TPanel(vComponent).Caption := caption2;
        inc(vi);
      end;
    end;
    if vStrList.Count>=9 then begin
      vDelta := (Panel5.Top)-(Panel6.Top+Panel6.Height);
      vTop := (vStrList.Count)*Panel3.Height + Panel3.Top;
      vComponent := TRdGroup.Create(ScrollBox1);
      //TControl(vComponent).Visible := False;
      TControl(vComponent).Parent := ScrollBox1;
      TControl(vComponent).SetBounds(Panel6.Left, vTop+vDelta,
Panel6.Width,
        Panel6.Height);
      TTrackGroup(vComponent).CreateButton(1);
    end;
  end;
  ScrollBox1.Visible := True;
  {for i:=ScrollBox1.ComponentCount-1 downto 0 do begin
    vComponent := ScrollBox1.Components[i];
    if vComponent is TControl then begin
      //TControl(vComponent).Visible := True;
    end;
  end;}

end;

procedure TFormQuest2.Button1Click(Sender: TObject);
var vStrList: TStringList;
  vCurrent: string;
  i, k: Integer;
  vTerus: Boolean;
```

```
begin
(*  vTerus := False;
  with FormMain do begin
    if MyTree.SelectedElement = nil then begin
      MyTree.SelectedElement := MyTree.RootElement;
    end;
    if MyTree.SelectedElement <> nil then begin
      FormPreferensi.NamaData := MyTree.SelectedElement.DataName;
      if MyTree.SelectedElement.FirstChild <> nil then begin
        vTerus := True;
      end else begin
        if (MyTree.SelectedElement.Parent <> nil) and
          (MyTree.RootAlternative.FirstChild <> nil) then begin
          vTerus := True;
        end;
      end;
    end;
  end;
  if not vTerus then exit;
  vStrList := TStringList.Create;
  LockWindowUpdate(ScrollBox1.Handle);
  try
    FormMain.CompareBckClick(Self);
    Label1.Caption := FormPreferensi.ComboBox2.Text;
    {if FormPreferensi.ComboBox2.ItemIndex>=0 then begin
      Label1.Caption :=
FormPreferensi.ComboBox2.Items[FormPreferensi.ComboBox2.ItemIndex];
    end else begin
      Label1.Caption := '';
    end;}

    Screen.Cursor := crHourGlass;
    for k:=0 to
FormPreferensi.DataPreferensi.ListNamaKriteria.Count-2 do begin
      vCurrent := FormPreferensi.DataPreferensi.ListNamaKriteria[k];
      for i:=k+1 to
FormPreferensi.DataPreferensi.ListNamaKriteria.Count-1 do begin

vStrList.Add(vCurrent+','+FormPreferensi.DataPreferensi.ListNamaKrit
eria[i]);
      end;
    end;
    CreateCompareObj(StatusEntry, vCurrent, vStrList);
    {if StatusEntry = 0 then begin
      StatusEntry := 1;
      Button1.Caption := 'Real Values';
    end else if StatusEntry = 1 then begin
      StatusEntry := 2;
      Button1.Caption := 'Real Track';
    end else begin
      StatusEntry := 0;
      Button1.Caption := 'Integer Values';
    end;}
  finally
    LockWindowUpdate(0);
    Screen.Cursor := crDefault;
```

```
    vStrList.Free;
  end;
  {ScrollBox1.Refresh;}
  *)
end;

procedure TFormQuest2.BtnPrevClick(Sender: TObject);
begin
(*  if FormPreferensi.ComboBox2.ItemIndex>0 then begin
    FormPreferensi.Button4Click(nil);
    {if StatusEntry=0 then begin
      StatusEntry := 1;
    end else begin
      StatusEntry := 0;
    end;}
    Button1Click(nil);
  end;*)
end;

procedure TFormQuest2.BtnNextClick(Sender: TObject);
begin
(*  if
FormPreferensi.ComboBox2.ItemIndex<FormPreferensi.ComboBox2.Items.Co
unt-1 then begin
    FormPreferensi.Button2Click(nil);
    {if StatusEntry=0 then begin
      StatusEntry := 1;
    end else begin
      StatusEntry := 0;
    end;}
    Button1Click(nil);
    if Assigned(Refreshing) then begin
      Refreshing(Self);
    end;
  end; *)
end;

procedure TFormQuest2.Edit1Change(Sender: TObject);
var vProcess: Boolean;
  vNilai: Extended;
  vTanda: Integer;
  vIndex: Integer;
  //pData: pTDataRecord;
  vId: string;
  vFIeld: TKFieldObject;
  i: Integer;
begin
  {}
  if FDontChange then exit;
  vTanda := 0;
  vIndex := 0;
  vNilai := 1;
  vProcess := False;
  if Sender is TEditGroup then begin
    vIndex := TEditGroup(Sender).Tag;
    vNilai := TEditGroup(Sender).FNilai;
```

```
    vTanda := TEditGroup(Sender).FTanda;
    vProcess := True;
  end else if Sender is TRdGroup then begin
    vIndex := TRdGroup(Sender).Tag;
    vNilai := TRdGroup(Sender).FNilai;
    vTanda := TRdGroup(Sender).FTanda;
    vProcess := True;
  end else if Sender is TTrackGroup then begin
    vIndex := TTrackGroup (Sender).Tag;
    vNilai := TTrackGroup (Sender).FNilai;
    vTanda := TTrackGroup (Sender).FTanda;
    vProcess := True;
  end;
  if vProcess then begin
    if (vNilai < 1) or (vNilai>9) then begin
      ShowMessage('Data harus diantara 1 dan 9');
      vProcess := False;
    end;
  end;
  if vProcess then begin
    FDataUpdated := True;
    for i:=0 to CurrentCrit.FieldCount-1 do begin
      vField := CurrentCrit[i];
      if vField.TagValue3<>0 then begin
        FIndepChanged := True;
        vField.TagValue3 := 2;
      end;
    end;
    vId := FItemList[vIndex];
    FDataStru.SetMatrixData(FCurrentCrit, vId, vTanda, vNilai);
    CalculatingData;
    DrawGrid1.Refresh;
    {pData := FormPreferensi.DataPreferensi.GetData(vIndex, 0);
    if pData <> nil then begin
      pData^.Nilai := vNilai;
      pData^.Tanda := vTanda;
    end;}
    {if Assigned(DataChange) then begin
      DataChange(Sender);
    end;}
    ShowUpdateInfo;
  end;
end;

procedure TFormQuest2.GetData(indeks: Integer; var Tanda: Integer;
  var Nilai: Extended);
var vId: string;
begin
  Tanda := 0;
  Nilai := 1;
  try
    vId := FItemList[Indeks];
    FDataStru.GetMatrixData(FCurrentCrit, vId, Tanda, Nilai);
  except
  end;
end;
```

```
procedure TFormQuest2.Button4Click(Sender: TObject);
begin
  FormGuide.Show;
end;

procedure TFormQuest2.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  Label1.Caption := IntToStr(Ord(Key));
end;

procedure TFormQuest2.SetCurrentCrit(const Value: TKBlockObject);
begin
  FCurrentCrit := Value;
end;

procedure TFormQuest2.SetDataStru(const Value: TRDataStru);
begin
  FDataStru := Value;
end;

procedure TFormQuest2.BuildEntry;
var vStrList, vStrList1: TStringList;
  i, j: Integer;
begin
  //LockWindowUpdate(ScrollBox1.Handle);
  FDontChange := True;
  try
    Edit1.Text := '';
    try
      if FCurrentCrit.OwnerField.Owner.OwnerField<>nil then begin
        Edit1.Text :=
FCurrentCrit.OwnerField.Owner.OwnerField.FieldName;
      end else begin
        Edit1.Text := 'Goal';
      end;
    except
    end;
    if (FDataStru<>nil) and (FCurrentCrit<>nil) then begin
      vStrList := TStringList.Create;
      vStrList1 := TStringList.Create;
      try
        if FCurrentCrit.FieldCount>0 then begin
          for i:=0 to FCurrentCrit.FieldCount-1 do begin
            vStrList.Add(FCurrentCrit[i].FieldName);
          end;
        end else begin
          for i:=0 to FDataStru.AltList.FieldCount-1 do begin
            vStrList.Add(FDataStru.AltList[i].FieldName);
          end;
        end;
        for i:=0 to vStrList.Count-2 do begin
          for j := i+1 to vStrList.COunt-1 do begin
            vStrList1.Add(vStrList[i]+','+vStrList[j]);
          end;
        end;
```

```
        FCritList.Text := vStrList.Text;
        FItemList.Text := vStrList1.Text;
        DrawGrid1.RowCount := vStrList.Count+1;
        DrawGrid1.ColCount := vStrList.Count+3;
        CreateCompareObj(FStatus, 'xx', vStrList1);
        CalculatingData;
        DrawGrid1.Refresh;
      finally
        vStrList.Free;
        vStrList1.Free;
      end;
    end;
  finally
    FDontChange := False;
    //LockWindowUpdate(0);
  end;
end;

procedure TFormQuest2.SetStatus(const Value: Integer);
begin
  FStatus := Value;
end;

procedure TFormQuest2.SetData(indeks, Tanda: Integer; Nilai:
Extended);
var vId: string;
begin
  try
    vId := FItemList[Indeks];
    FDataStru.SetMatrixData(FCurrentCrit, vId, Tanda, Nilai);
  except
  end;
end;

procedure TFormQuest2.CalculatingData;
begin
  if FCurrentCrit.FieldCount = 0 then begin
    FDataStru.HitungAlternative(FCurrentCrit, FTemporerArray);
  end else begin
    FDataStru.HitungKriteria(FCurrentCrit, FTemporerArray);
  end;
end;

procedure TFormQuest2.DrawGrid1DrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
var vText: string;
begin
  vText := '';
  if ARow =0 then begin
    if ACol=0 then begin
    end else if ACol=DrawGrid1.ColCount-2 then begin
      vText := 'Rough Weight';
    end else if ACol=DrawGrid1.ColCount-1 then begin
      vText := 'Real Weight';
    end else begin
      vText := FCritList[ACol-1];
```

```
    end;
  end else begin
    if ACol=0 then begin
      vText := FCritList[ARow-1];
    end else begin
      vText := FormatFloat('0.#####', FTemporerArray[ACol-1, ARow-
1]);
    end;
  end;
  TextDraw(DrawGrid1.Canvas, vText,
DT_CENTER+DT_VCENTER+DT_SINGLELINE,
    Rect);
end;

procedure TFormQuest2.Button5Click(Sender: TObject);
begin
  FormValidasi.ArrayValidasi := FTemporerArray;
  FormValidasi.CriteriaList.Text := FCritList.Text;
  FormValidasi.PrepareData;
  FormValidasi.Edit1.Text := Edit1.Text;
  FormValidasi.ShowModal;
end;

procedure TFormQuest2.SetDataUpdated(const Value: Boolean);
begin
  FDataUpdated := Value;
end;

procedure TFormQuest2.SetIndepChanged(const Value: Boolean);
begin
  FIndepChanged := Value;
end;

procedure TFormQuest2.ShowUpdateInfo;
var vStr: string;
begin
  vStr := '';
  if FIndepChanged then begin
    vStr := vStr+'Modification make interdependency changed';
  end;
  StatusBar1.panels[0].Text := vStr;
end;

procedure TFormQuest2.FormShow(Sender: TObject);
begin
  FDataUpdated := False;
  FIndepChanged := False;
end;

end.
```

```
unit GlobalUnit;

interface

uses sysutils, windows, classes, graphics;

const MAX_CRIT = 30;
type
  TArrayCriteria = array[0..MAX_CRIT, 0..MAX_CRIT] of Extended;
  TVectorCriteria = array[0..MAX_CRIT] of Extended;

procedure TextDraw(Canvas: TCanvas; S: string; Style: Word; var
xRect: TRect);
function HitungAkar(Nilai, fAkar: Extended): Extended;

implementation

procedure TextDraw(Canvas: TCanvas; S: string; Style: Word; var
xRect: TRect);
var xchar: Array[0..255] of Char;
begin
  StrPCopy(xchar, S);
  DrawText(Canvas.Handle, xchar, -1, xRect, Style);
end;

function HitungAkar(Nilai, fAkar: Extended): Extended;
var vValue: Extended;
begin
  if Nilai<=0 then begin
    result := 0;
  end else begin
    if fAkar<=0 then begin
      result := 1;
    end else begin
      vValue := 1/fAkar;
      result := exp(vValue*ln(Nilai));
    end;
  end
end;


end.
```

```
unit Fvalid;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, StdCtrls, ExtCtrls, Grids, GlobalUnit;

type
  TFormValidasi = class(TForm)
    Panel4: TPanel;
    Panel3: TPanel;
    Label3: TLabel;
    ComboBox2: TComboBox;
    Label1: TLabel;
    Panel1: TPanel;
    Button1: TButton;
    Edit1: TEdit;
    DrawGrid1: TDrawGrid;
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure StringGrid1DrawCellx(Sender: TObject; Col, Row:
Longint;
      xRect: TRect; State: TGridDrawState);
    procedure Button1Click(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure ComboBox2Change(Sender: TObject);
    procedure DrawGrid1DrawCell(Sender: TObject; ACol, ARow:
Integer;
      Rect: TRect; State: TGridDrawState);
  private
    { Private declarations }
    StringFormat: string;
    FCriteriaList: TStrings;
    function GetJumlahElement: Integer;
  public
    { Public declarations }
    ArrayValidasi: TArrayCriteria;
    VektorValidasi, BobotKasar, BobotNyata: TVectorCriteria;
    //ParentValidasi: TTreeElement;
    RasioKonsistensi: Extended;
    DataError: Boolean;
    //TreeObject: TTreeContainer;
    ListParent: TList;
    PrepareObject: TNotifyEvent;
    property CriteriaList: TStrings read FCriteriaList;
    property JumlahElement: Integer read GetJumlahElement;
    procedure HitungKonsistensi;
    procedure DrawKonsistensiValue;
    procedure PrepareData;
    procedure ShowKonsistensi;
    //procedure GetListParent(Element: TTreeElement; var Stop:
Boolean);
  end;
```

```
var
  FormValidasi: TFormValidasi;

implementation

{$R *.DFM}

procedure TFormValidasi.FormCreate(Sender: TObject);
var i, j: Integer;
begin
  ListParent := TList.Create;
  for i:=1 to 30 do begin
    for j:=1 to 30 do begin
      ArrayValidasi[i,j] := 1;
    end;
    BobotKasar[i] := 1;
    BobotNyata[i] := 1;
  end;
  StringFormat := Application.Title + ' ['+Caption+' - %s]';

{  ArrayValidasi[1, 2] := 5;
  ArrayValidasi[1, 3] := 6;
  ArrayValidasi[1, 4] := 7;
  ArrayValidasi[2, 3] := 4;
  ArrayValidasi[2, 4] := 6;
  ArrayValidasi[3, 4] := 4;}

  FCriteriaList := TStringList.Create;

end;

procedure TFormValidasi.FormShow(Sender: TObject);
{var vElement: TTreeElement;
  i, j: Integer;
  vTeks: string;
  vNilai: Extended;
  vValid: Boolean;
  vValue1, vValue2, vValue3: Extended;
  pData: pTDataKriteria;}
begin
(*  Caption := Format(StringFormat, [ComboBox2.Text]);
  StringGrid1.RowCount := 2;
  StringGrid1.ColCount := 2;
  StringGrid1.RowCount := JumlahElement+1;
  StringGrid1.ColCount := JumlahElement+1;
  if ParentValidasi<>nil then begin
    vElement := ParentValidasi.FirstChild;
    if vElement = nil then begin
      vElement := TreeObject.RootAlternative.FirstChild;
    end;
    for i:=1 to JumlahElement do begin
      if vElement <> nil then begin
        StringGrid1.Cells[i, 0] := vELement.DataName;
        StringGrid1.Cells[0, i] := vELement.DataName;
        vElement := vELement.NextSibling;
```

```
      end;
    end;
    StringGrid1.Cells[0, 0] := 'Criteria';
    StringGrid1.Cells[JumlahElement+1, 0] := 'Rough Weight';
    StringGrid1.Cells[JumlahElement+2, 0] := 'Real Weight';
  end;
  for i:=0 to JumlahElement-1 do begin
    for j:=0 to JumlahElement-1 do begin
      vValue1 := 1;
      vValue2 := 1;
      ArrayValidasi[i+1, j+1] := -1;
      pData := ParentValidasi.DataPreferensi.GetDataKriteria(i);
      StringGrid1.Cells[i+1, j+1] := '';
      DataError := True;
      if pData <> nil then begin
        vValue1 := pData^.NilaiRataRata;
        pData := ParentValidasi.DataPreferensi.GetDataKriteria(j);
        if pData<> nil then begin
          vValue2 := pData^.NilaiRataRata;
          vValue3 := vValue1/vValue2;
          if vValue3>9 then begin
            vValue3 := 9;
          end;
          if vValue3 <1/9 then begin
            vValue3 := 1/9;
          end;
          ArrayValidasi[i+1, j+1] := vValue3;
          DataError := False;
          {vTeks := FormatFloat('####0.####', vValue2/vValue1);
          StringGrid1.Cells[i+1, j+1] := vTeks;}
        end;
      end;
    end;
  end;
  for i:=0 to JumlahElement-1 do begin
    DataError := True;
    pData := ParentValidasi.DataPreferensi.GetDataKriteria(i);
    BobotKasar[i+1] := -1;
    BobotNyata[i+1] := -1;
    if pData<>nil then begin
      BobotKasar[i+1] := pData^.BobotKasar;
      BobotNyata[i+1] := pData^.BobotNyata;
      DataError := False;
    end;
  end;
  if not DataError then begin
    HitungKonsistensi;
  end;
  DrawKonsistensiValue; *)
end;
```

```
procedure TFormValidasi.StringGrid1DrawCellx(Sender: TObject; Col,
  Row: Longint; xRect: TRect; State: TGridDrawState);
var vTeks: string;
  vNilai: Extended;
begin
  (*
  vTeks := '';
  vNilai := -1;
  if (Row >0) and (Col>0) then begin
    if Col<=JumlahElement then begin
      vNilai := ArrayValidasi[Row, Col];
      {vTeks := FormatFloat('###0.###', ArrayValidasi[Row, Col]);}
    end else if Col=JumlahElement+1 then begin
      vNilai := BobotKasar[Row];
      {vTeks := FormatFloat('###0.###', BobotKasar[Row]);}
    end else if Col=JumlahElement+2 then begin
      vNilai := BobotNyata[Row];
      {vTeks := FormatFloat('###0.###', BobotNyata[Row]);}
    end else if Col=JumlahElement+3 then begin
      vNilai := VektorValidasi[Row];
      {vTeks := FormatFloat('###0.###', VektorValidasi[Row]);}
    end;
  end;
  if vNilai <=0 then begin
    vTeks := '';
  end else begin
    vTeks := FormatFloat('###0.####', vNilai);
  end;
  if vTeks <> '' then begin
    StringGrid1.Canvas.FillRect(xRect);
    TextDraw(StringGrid1.Canvas, vTeks, DT_LEFT, xRect);
  end;
  *)
end;

procedure TFormValidasi.DrawKonsistensiValue;
begin
  if DataError then begin
    Label1.Caption := 'Consistency Ratio : <Cannot Evaluated>';
  end else begin
    Label1.Caption := 'Consistency Ratio : '+FormatFloat('###0.###',
RasioKonsistensi);
  end;
  if RasioKonsistensi<=0.1 then begin
    Label1.Caption := Label1.Caption;
  end else begin
    Label1.Caption := Label1.Caption;
  end;
end;

procedure TFormValidasi.HitungKonsistensi;
var i, j: Integer;
  vNilai: Extended;
  vJumlah: Extended;
  vIndexRandom: Extended;
begin
```

```
  for i:=0 to JumlahElement-1 do begin
    BobotKasar[i] := ArrayValidasi[JumlahElement, i];
    BobotNyata[i] := ArrayValidasi[JumlahElement+1, i];
  end;

  vJumlah := 0;
  for i:= 0 to JumlahElement-1 do begin
    VektorValidasi[i] := 0;
    for j := 0 to JumlahElement-1 do begin
      VektorValidasi[i] := VektorValidasi[i]+ArrayValidasi[j,
i]*BobotNyata[j];
    end;
    vEktorValidasi[i] := VektorValidasi[i]/BobotNyata[i];
    vJumlah := vJumlah+VektorValidasi[i];
  end;
  vJumlah := vJumlah/JumlahElement; {Eigen Value Max}
  vJumlah := (vJumlah-JumlahElement)/(JumlahElement-1); {CI}
  Case JumlahElement of
    1, 2: vIndexRandom := 0;
    3: vIndexRandom := 0.58;
    4: vIndexRandom := 0.9;
    5: vIndexRandom := 1.12;
    6: vIndexRandom := 1.24;
    7: vIndexRandom := 1.32;
    8: vIndexRandom := 1.41;
    9: vIndexRandom := 1.45;
    10: vIndexRandom := 1.49;
    11: vIndexRandom := 1.51;
    12: vIndexRandom := 1.48;
    13: vIndexRandom := 1.56;
    14: vIndexRandom := 1.57;
    15: vIndexRandom := 1.59;
  else
    vIndexRandom := 0.01;
  end;
  if vIndexRandom = 0 then begin
    RasioKonsistensi := 0;
  end else begin
    RasioKonsistensi := vJumlah/vIndexRandom;
  end;
  ShowKonsistensi;
end;

procedure TFormValidasi.Button1Click(Sender: TObject);
begin
  Close;
end;

procedure TFormValidasi.FormDestroy(Sender: TObject);
begin
  ListParent.Free;
  FCriteriaList.Free;
end;
```

```
{procedure TFormValidasi.GetListParent(Element: TTreeElement; var
Stop: Boolean);
begin
  ListParent.Add(Element);
  ComboBox2.Items.Add(Element.DataName);
end;
}
procedure TFormValidasi.ComboBox2Change(Sender: TObject);
begin
{  if Assigned(PrepareObject) then begin
    TreeObject.SelectedElement.UpDateData;
    TreeObject.UpdatePosition;
    TreeObject.Invalidate;
    TreeObject.SelectedElement := ListParent[ComboBox2.ItemIndex];
    PrepareObject(nil);
    FormShow(nil);
  end;}
end;


function TFormValidasi.GetJumlahElement: Integer;
begin
  result := FCriteriaList.Count;
end;

procedure TFormValidasi.DrawGrid1DrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
var vText: string;
begin
  vText := '';
  if ARow =0 then begin
    if ACol=0 then begin
    end else begin
      vText := FCriteriaList[ACol-1];
    end;
  end else begin
    if ACol=0 then begin
      vText := FCriteriaList[ARow-1];
    end else begin
      vText := FormatFloat('0.#####', ArrayValidasi[ACol-1, ARow-
1]);
    end;
  end;
  TextDraw(DrawGrid1.Canvas, vText,
DT_CENTER+DT_VCENTER+DT_SINGLELINE,
    Rect);
end;

procedure TFormValidasi.PrepareData;
begin
  DrawGrid1.ColCount := JumlahElement+1;
  DrawGrid1.RowCount := JumlahElement+1;
  HitungKonsistensi;
end;
```

```
procedure TFormValidasi.ShowKonsistensi;
begin
  Label1.Caption := 'Konsistensi : '+FormatFloat('0.####',
RasioKonsistensi);
end;

end.
```

```
unit Fguide;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, StdCtrls, ExtCtrls;

type
  TFormGuide = class(TForm)
    Button1: TButton;
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel5: TPanel;
    Panel6: TPanel;
    Panel7: TPanel;
    Panel8: TPanel;
    Panel9: TPanel;
    Panel10: TPanel;
    Panel11: TPanel;
    Panel12: TPanel;
    Panel13: TPanel;
    Panel14: TPanel;
    Panel15: TPanel;
    Panel18: TPanel;
    Panel19: TPanel;
    Label1: TLabel;
    CheckBox1: TCheckBox;
    procedure Button1Click(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormGuide: TFormGuide;

implementation

{$R *.DFM}

procedure TFormGuide.Button1Click(Sender: TObject);
begin
  Close;
end;
```

```
procedure TFormGuide.CheckBox1Click(Sender: TObject);
begin
  if CheckBox1.Checked then begin
    FormStyle := fsStayOnTop;
  end else begin
    FormStyle := fsNormal;
  end;
end;

end.
```

```
unit Findep;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, ExtCtrls, Spnl, {DTree, SData, }Grids, StdCtrls,
FPanduan,
  KFieldObjects, GlobalUnit;

type
  TFormInterDependency = class(TForm)
    Panel2: TPanel;
    ScrollBox1: TScrollBox;
    Panel1: TPanel;
    Panel4: TPanel;
    Label3: TLabel;
    Panel6: TPanel;
    Label1: TLabel;
    Panel7: TPanel;
    Panel5: TPanel;
    Button1: TButton;
    Button2: TButton;
    BtnPrev: TButton;
    BtnNext: TButton;
    PanelOwner: TPanel;
    Edit1: TEdit;
    Panel8: TPanel;
    Splitter1: TSplitter;
    DrawGrid1: TDrawGrid;
    StringGrid2: TStringGrid;
    StringGrid1: TStringGrid;
    Panel3: TPanel;
    Panel9: TPanel;
    Splitter2: TSplitter;
    Splitter3: TSplitter;
    procedure FormCreate(Sender: TObject);
    procedure Panel1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure FormDestroy(Sender: TObject);
    procedure Shape1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape1MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape1MouseMove(Sender: TObject; Shift: TShiftState;
X,
      Y: Integer);
    procedure FormShow(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure StringGrid1DrawCell(Sender: TObject; Col, Row:
Longint;
      Rect: TRect; State: TGridDrawState);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure ComboBox2Change(Sender: TObject);
    procedure Button2Click(Sender: TObject);
```

```
    procedure FormCloseQuery(Sender: TObject; var CanClose:
Boolean);
    procedure BtnPrevClick(Sender: TObject);
    procedure BtnNextClick(Sender: TObject);
    procedure DrawGrid1DrawCell(Sender: TObject; ACol, ARow:
Integer;
      Rect: TRect; State: TGridDrawState);
    procedure StringGrid2DrawCell(Sender: TObject; ACol, ARow:
Integer;
      Rect: TRect; State: TGridDrawState);
  private
    { Private declarations }
    FLastX, FLastY: Integer;
    MovingShape: TShape;
    ListAltData: TList;
    FUpdateData: Boolean;
    StringFormat: string;
    {procedure CallBackListKorelasi(Element: TTreeElement;
      var Stop: Boolean);}
    FCritParent: TKBlockObject;
    FJumlahLuas: Extended;
    FJumlahLuasIndep: Extended;
    FIndependentList: TStrings;
    FListSegment: TList;
    procedure SetCritParent(const Value: TKBlockObject);
  public
    { Public declarations }
    MyFormat: string;
    MyArrayColor: array[1..10] of TColor;
    MyBackColor: array[1..10] of TColor;
    //ParentElement: TTreeElement;
    //ParentAlternative: TTreeElement;
    ListKrit: TList;
    ListRect: TList;
    ListParent: TList;
    //TreeObject: TTreeContainer;
    PrepareObject: TNotifyEvent;
    procedure ClearList;
    //procedure AddKriteria(Elemen: TTreeElement);
    procedure MyDrawPanel(Sender: TObject; Canvas: TCanvas;
ObjBound: TRect);
    procedure InvalidateAll;
    procedure UpdateGrid;
    //procedure ProsesKorelasi(Mode: Integer);
    //function HitungKorelasi(Data1, Data2: TTreeElement): Extended;
    function DisplayAllRect(Target:TStrings; TinggiAwal: Integer;
      DisplayFirstLevel: Boolean): Integer;
    //procedure GetListParent(Element: TTreeElement; var Stop:
Boolean);
    property CritParent: TKBlockObject read FCritParent write
SetCritParent;
    procedure HitungIndep(ListRect: TList; Parent: TObject);
  end;
```

```
var
  FormInterDependency: TFormInterDependency;

function CekId(id1, id2: string): Boolean;
procedure ClearListRect(ListRect: TList);

implementation

{$R *.DFM}

type

  TXData = record
    value1, value2: Extended;
  end;
  pTXData = ^TXData;

  TDataRect = record
    id: string;
    RectData: TRect;
    Luas: Extended;
    BobotLuas: Extended;
    JmlBobotAwal: Extended;
  end;
  pTDataRect = ^TDataRect;


procedure TFormInterDependency.FormCreate(Sender: TObject);
begin
  //ParentAlternative := nil;
  ListKrit := TList.Create;
  ListRect := TList.Create;
  MovingShape := nil;
  MyArrayColor[1] := $FFFFFF;
  MyArrayColor[2] := $DDDDDD;
  MyArrayColor[3] := $BBBBBB;
  MyArrayColor[4] := $999999;
  MyArrayColor[5] := $777777;
  MyArrayColor[6] := $555555;
  MyArrayColor[7] := $444444;
  MyArrayColor[8] := $333333;
  MyArrayColor[9] := $111111;
  MyArrayColor[10] := $000000;
  MyFormat := '0.0###';

  MyBackColor[1] := clYellow+$440000;
  MyBackColor[2] := clAqua+$000044;
  MyBackColor[3] := clBlue+$009999;
  MyBackColor[4] := clLime+$990099;
  MyBackColor[5] := clRed+$999900;
  MyBackColor[6] := clFuchsia+$009900;
  MyBackColor[7] := clBlue+$004444;
  MyBackColor[8] := clLime+$440044;
  MyBackColor[9] := clRed+$444400;
  MyBackColor[10] := clFuchsia+$004400;
{  MyFormat := '0.00E+0##';}
```

```
  StringFormat := Application.Title + ' ['+Caption+' - %s]';
  ListParent := TList.Create;
  FIndependentList := TStringLIst.Create;
  FListSegment := TList.Create;
  StringGrid1.ColWidths[0] := 120;
  StringGrid2.ColWidths[0] := 120;
  DrawGrid1.ColWidths[0] := 120;
end;

procedure TFormInterDependency.FormDestroy(Sender: TObject);
var pData: pTDataRect;
  i: Integer;
begin
  ClearList;
  ListKrit.Destroy;
  ClearListRect(ListRect);
  ListRect.Destroy;
  ListParent.Free;
  FIndependentList.Free;
  for i:=0 to FListSegment.count-1 do begin
    pData := FListSegment[i];
    Dispose(pData);
  end;
  FListSegment.Free;
end;

{procedure TFormInterDependency.AddKriteria(Elemen: TTreeElement);
var MyPanel: TDemiPanel;
  vJml: Integer;
  vLebar, vTinggi: Integer;
  Rasio: Extended;
begin
  vLebar := CLebarIndep;
  vTinggi := CTinggiIndep;
  vJml := ListKrit.Count;
  MyPanel := TDemiPanel.Create(Self);
  MyPanel.Caption := Elemen.DataName;
  MyPanel.Parent := Panel1.Parent;
  Rasio := HitungAkar(Elemen.PreferensiLokal, 2);
  MyPanel.Width := Round(vLebar * Rasio);
  MyPanel.Height := Round(vTinggi*Rasio);
  if not ((Elemen.IndepRect.Left=0) and (Elemen.IndepRect.Top=0)
    and (Elemen.IndepRect.Right=0) and (Elemen.IndepRect.Bottom=0))
then begin
    MyPanel.Left := Elemen.IndepRect.Left;
    MyPanel.Top := Elemen.IndepRect.Top;
  end else begin
    MyPanel.Top := vJml * (Panel1.Height+5)+Panel1.Top;
    MyPanel.Left := Panel1.Left;
  end;
  Elemen.IndepRect := MyPanel.BoundsRect;
  MyPanel.AbsBounds := MyPanel.BoundsRect;
  MyPanel.Data := Elemen;
  MyPanel.ShowHint := True;
  if ListKrit.Count+1<=10 then begin
    MyPanel.Color := MyBackColor[ListKrit.Count+1];
```

```
  end else begin
    MyPanel.Color := clWhite;
  end;
  MyPanel.Hint := MyPanel.Caption;
  MyPanel.OnMouseDown := Panel1MouseDown;
  MyPanel.OnDraw := MyDrawPanel;
  ListKrit.Add(MyPanel);
end;
}
procedure TFormInterDependency.Panel1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var //pData: TTreeElement;
  i: Integer;
  vcx, vcy: Integer;
  var vField: TKFIeldObject;
begin
  ReleaseCapture;
  TControl(Sender).perform(WM_SYSCOMMAND, $F012, 0);
  ScrollBox1.AutoScroll := False;
  ScrollBox1.AutoScroll := True;
  ScrollBox1.ScrollInView(TControl(Sender));
  vcx := TControl(Sender).Left;
  vcy := TControl(Sender).Top;
  vField := FCritParent[TControl(Sender).Tag];
  vFIeld.TagValue1 := vcx;
  vFIeld.TagValue2 := vcy;
  vFIeld.TagValue3 := 1;

  FUpdateData := True;
  HitungIndep(ListRect, nil);
  InvalidateAll;
  for i:=0 to ScrollBox1.ControlCount-1 do begin
    ScrollBox1.Controls[i].Refresh;
  end;
  ScrollBox1.Refresh;
  StringGrid1.Refresh;
  DrawGrid1.Refresh;
  StringGrid2.Refresh;
{  for i:=0 to ListKrit.Count-1 do begin
    pData := TDemiPanel(ListKrit[i]).Data;
    if pData <> nil then begin
      pData.IndepRect := TDemiPanel(ListKrit[i]).BoundsRect;
    end;
    HitungIndep(ListRect, ParentElement);
    InvalidateAll;
  end;}
end;
```

```
procedure TFormInterDependency.ClearList;
var i: Integer;
  MyPanel: TPanel;
begin
  for i:=0 to ListKrit.Count-1 do begin
    MyPanel := ListKrit[i];
    MyPanel.Free;
  end;
  ListKrit.Clear;
end;

procedure TFormInterDependency.Shape1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  MovingShape := TShape(Sender);
  FLastX := X;
  FLastY := Y;
end;

procedure TFormInterDependency.Shape1MouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  if MovingShape = Sender then begin
    MovingShape := nil;
  end;
end;

procedure TFormInterDependency.Shape1MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
var dX, dY: Integer;
begin
  if MovingShape = sender then begin
    dX := FLastX-X;
    dY := FLastY-Y;
    MovingShape.Left := MovingShape.Left-dX;
    MovingShape.Top := MovingShape.Top-dY;
  end;
end;

procedure TFormInterDependency.FormShow(Sender: TObject);
var i: Integer;
  //pElement: TTreeElement;
begin
  exit;
  FUpdateData := False;
  ScrollBox1.VertScrollBar.Position := 0;
  ClearList;
{  if ParentElement <> nil then begin
    pElement := ParentElement.FirstChild;
    while pElement <> nil do begin
      AddKriteria(pElement);
      pElement := pElement.NextSibling;
    end;
    HitungIndep(ListRect, ParentElement);
  end;}
  StringGrid1.ColCount := 3;
```

```
  StringGrid1.RowCount := ListKrit.Count+1;
  StringGrid2.ColCount := 3;
  DrawGrid1.ColCount := 2;
  {if ParentElement <> nil then begin
    StringGrid1.Cells[0, 0] := ParentElement.DataName;
  end else begin
    StringGrid1.Cells[0, 0] := '';
  end;}
  StringGrid1.Cells[1, 0] := 'Starting Weight';
  StringGrid1.Cells[2, 0] := 'Real Weight';
  UpDateGrid;
    //ProsesKorelasi(0);
{  if Sender <> nil then begin
    ProsesKorelasi(0);
  end;}
  //Caption := Format(StringFormat, [ParentElement.DataName]);
end;

function TFormInterDependency.DisplayAllRect(Target:TStrings;
TinggiAwal: Integer;
  DisplayFirstLevel: Boolean): Integer;
var i: Integer;
  //pData: pTDataRect;
  vTeks: string;
  vColId: Integer;
  vColor: TColor;
  xRect: TRect;
begin
  result := 0;
  for i:=0 to ListRect.Count-1 do begin
    //pData := ListRect[i];
    {vColId := Length(pData^.Id);
    if (vColId>0) and (vColId<=High(MyArrayColor)) then begin
      vColor := MyArrayColor[vColId];
    end else begin
      vColor := clWhite;
    end;
    if ((vColId <> 1) or (DisplayFirstLevel)) then begin
      xRect := pData^.RectData;
      vTeks := '';
      vTeks := vTeks+'@RT'+IntToStr(xRect.Left)+','+
        IntToStr(xRect.Top+TinggiAwal)+','+
        IntToStr(xRect.Right)+','+
        IntToStr(xRect.Bottom+TinggiAwal)+','+IntToStr(vColor)+',#';
      Target.Add(vTeks);
      result := result+1;
    end;}
  end;
end;
```

```
procedure TFormInterDependency.MyDrawPanel(Sender: TObject; Canvas:
TCanvas; ObjBound: TRect);
var i: Integer;
  pData: pTDataRect;
  vId: string;
  xRect: TRect;
  vTeks: string;
begin
  xRect := ScrollBox1.ClientRect;
  Canvas.Rectangle(0, 0, ObjBound.Right-ObjBound.Left,
ObjBound.Bottom-ObjBound.Top);
  xRect := ObjBound;

  //if TDemiPanel(Sender).Data<>nil then begin
    vId := TDemiPanel(Sender).RectId;
    for i:=0 to ListRect.Count-1 do begin
      if vId = pTDataRect(ListRect[i])^.Id then begin
        xRect := pTDataRect(ListRect[i])^.RectData;
      end;
    end;
    for i:=0 to ListRect.Count-1 do begin
      pData := ListRect[i];
      if CekId(vId, pData^.Id) then begin
        if (Length(pData^.Id)>0) and (Length(pData^.Id)<=10) then
begin
          if Length(pData^.Id)=1 then begin
            Canvas.Brush.Color := TDemiPanel(Sender).Color;
          end else begin
            Canvas.Brush.Color := MyArrayColor[Length(pData^.Id)];
          end;
        end else begin
          Canvas.Brush.Color := MyArrayColor[10];
        end;
        Canvas.Rectangle(pData^.RectData.Left-xRect.Left,
          pData^.RectData.Top-xRect.Top,
          pData^.RectData.Right-xRect.Left,
          pData^.RectData.Bottom-xRect.Top);
      end;
    end;
{  try
    pData := ListRect[TControl(Sender).Tag];
    vTeks := pData^.Id;
    xRect := Rect(0, 0, TControl(Sender).Width,
TControl(Sender).Height);
    Canvas.Font.Color := clBlack;
    TextDraw(Canvas, vTeks, DT_CENTER+DT_VCENTER+DT_SINGLELINE,
xRect);
  except
  end;}
  //end;
end;
```

```
procedure TFormInterDependency.InvalidateAll;
var i: Integer;
  vRect: TRect;
begin
  for i:=0 to ListKrit.Count-1 do begin
    vRect := TDemiPanel(ListKrit[i]).BoundsRect;
    TDemiPanel(ListKrit[i]).Invalidate;
  end;
  updateGrid;
end;

procedure TFormInterDependency.UpdateGrid;
var i: Integer;
  //pElement: TTreeElement;
  vShow: Boolean;
begin
{  if ParentElement <> nil then begin
    pElement := ParentElement.FirstChild;
    i:=1;
    vShow := True;
    while pElement <> nil do begin
      StringGrid1.Cells[0, i] := pElement.DataName;
      StringGrid1.Cells[1, i] := FormatFloat(MyFormat,
pElement.PreferensiLokal);
      StringGrid1.Cells[2, i] := FormatFloat(MyFormat,
pElement.IndepValue*pElement.PreferensiLokal);
      if (pElement.IndepValue = 0) and vShow then begin
        vShow := False;
        ShowMessage('Data bertumpuk 100 % tidak dibenarkan.');
      end;
      i:=i+1;
      pElement := pElement.NextSibling;
    end;
  end;}
end;

(*procedure TFormInterDependency.CallBackListKorelasi(Element:
TTreeElement;
  var Stop: Boolean);
var i: Integer;
  pDataKrit: pTDataKriteria;
  pXData: ^TXData;
begin
  if Element.FirstChild = nil then begin
    if ListAltData=nil then begin
      Stop := True;
    end else begin
      for i:=0 to ListAltData.Count-1 do begin
        pXData := ListAltData[i];
        pDataKrit := Element.DataPreferensi.GetDataKriteria(i);
        if pDataKrit = nil then begin
          pXData^.Value1 := pXData^.Value1+
            {Element.Preferensi}1/ListAltData.Count;
        end else begin
          pXData^.Value1 := pXData^.Value1+
            {Element.Preferensi*}pDataKrit^.BobotNyata;
```

```
        end;
      end;
    end;
  end;
end;
*)
(*function TFormInterDependency.HitungKorelasi(Data1, Data2:
TTreeElement): Extended;
var pData: TTreeElement;
  MyList: TList;
  pXData: ^TXData;
  i: Integer;
  Stop: Boolean;
  ZC1, ZC2, ZC1C2, ZC1_2, ZC2_2: Extended;
  n: Extended;
  vTotal: Extended;
begin
  if ParentAlternative = nil then begin
    result := 0;
  end else begin
    result := 0;
    MyList := TList.Create;
    pData := ParentAlternative.FirstChild;
    while pData<>nil do begin
      New(pXData);
      pXData^.Value1 := 0;
      pXData^.Value2 := 0;
      MyList.Add(pXData);
      pData := pData.NextSibling;
    end;
    ListAltData := MyList;
    Stop := False;
    Data1.Trace(CallBackListKorelasi, Stop);
    for i:=0 to MyList.Count-1 do begin
      pXData := MyList[i];
      pXData^.Value2 := pXData^.Value1;
      pXData^.Value1 := 0;
    end;
    Stop := False;
    Data2.Trace(CallBackListKorelasi, Stop);
    n := MyList.Count;
    ZC1 := 0;
    ZC2 := 0;
    ZC1C2 := 0;
    ZC1_2 := 0;
    ZC2_2:=0;
    for i:=0 to MyList.Count-1 do begin
      pXData := MyList[i];
      ZC1 := ZC1+pXData^.Value1;
      ZC2 := ZC2+pXData^.Value2;
      ZC1C2 := ZC1C2+pXData^.Value1*pXData^.Value2;
      ZC1_2 := ZC1_2+pXData^.Value1*pXData^.Value1;
      ZC2_2 := ZC2_2+pXData^.Value2*pXData^.Value2;
    end;
    vTotal := (n*ZC1_2-ZC1*ZC1)*(n*ZC2_2-ZC2*ZC2);
    if vTotal=0 then begin
```

```
      result := 0;
    end else begin
      result := (n*ZC1C2-ZC1*ZC2)/sqrt(Abs(vTotal));
    end;
    ListAltData := nil;
    for i:=0 to MyList.Count-1 do begin
      pXData := MyList[i];
      Dispose(pXData);
    end;
    MyList.Destroy;
  end;
end;
*)
(*procedure TFormInterDependency.ProsesKorelasi(Mode: Integer);
var i, j: Integer;
  pElement: TTreeELement;
  MyList: TList;
  Hasil: Extended;
begin
{  Memo1.Lines.BeginUpdate;}
  if ParentElement<>nil then begin
    MyList := TList.Create;
    pElement := ParentElement.FirstChild;
    while pElement <> nil do begin
      MyList.Add(pElement);
      pElement := pElement.NextSibling;
    end;
    if Mode = 0 then begin
      FormPanduan.StringGrid2.ColCount := MyList.Count+1;
      FormPanduan.StringGrid2.RowCount := MyList.Count+1;
      FormPanduan.StringGrid2.Cells[0, 0] := 'Criteria';
      for i:=0 to MyList.Count-1 do begin
        FormPanduan.StringGrid2.Cells[i+1, i+1] := '1';
        FormPanduan.StringGrid2.Cells[0, i+1] :=
TTreeElement(MyList[i]).DataName;
        FormPanduan.StringGrid2.Cells[i+1, 0] :=
TTreeElement(MyList[i]).DataName;
        for j:=i+1 to MyList.Count-1 do begin
          Hasil := HitungKorelasi(MyList[i], MyList[j]);
          FormPanduan.StringGrid2.Cells[i+1, j+1] :=
FormatFloat(MyFormat, Hasil);
          FormPanduan.StringGrid2.Cells[j+1, i+1] :=
FormatFloat(MyFormat, Hasil);
        end;
      end;
    end else begin
    end;
    MyList.Destroy;
  end;
{  Memo1.Lines.EndUpdate;}
end;
*)

procedure TFormInterDependency.Button1Click(Sender: TObject);
begin
  if FormPanduan.CheckBox1.Checked then begin
```

```
    FormPanduan.FormStyle := fsNormal;
    FormPanduan.FormStyle := fsStayOnTop;
  end else begin
    FormPanduan.FormStyle := fsNormal;
  end;
  FormPanduan.Show;
end;

procedure TFormInterDependency.StringGrid1DrawCell(Sender: TObject;
Col,
  Row: Longint; Rect: TRect; State: TGridDrawState);
var vTeks: string;
  vField: TKFieldObject;
begin

  vTeks := '';
  if Row = 0 then begin
    case col of
      0: begin
        try
          vTeks :=
FCritParent.OwnerField.Owner.OwnerField.FieldName;
        except
          vTeks := 'Goal';
        end;
      end;
      1: begin
        vTeks := 'Starting Weight';
      end;
      2: begin
        vTeks := 'Real Weight';
      end;
    end;
  end else begin
    case Col of
      0: begin
        StringGrid1.Canvas.Brush.Color := MyBackColor[Row];
        StringGrid1.Canvas.FillRect(Rect);
        try
          vTeks := chr(Row+64)+' '+FCritParent[Row-1].FieldName;
        except
        end;
      end;
      1: begin
        try
          vField := TKBlockField(FCritParent[Row-
1]).BlockData.FieldByName('Criteria');
          vTeks := FormatFloat('0.####',
TKBlockField(vField).BlockData.TempReal2);
        except
        end;
      end;
      2: begin
        vField := TKBlockField(FCritParent[Row-
1]).BlockData.FieldByName('Criteria');
```

```
        vTeks := FormatFloat('0.####',
TKBlockField(vField).BlockData.TempReal4);
      end;
    end;
  end;
  StringGrid1.Canvas.TextOut(Rect.Left+1, Rect.Top+1, vTeks);
end;

procedure TFormInterDependency.FormClose(Sender: TObject;
  var Action: TCloseAction);
{var pData: TTreeElement;
  i: Integer;}
begin
{  ScrollBox1.VertScrollBar.Position := 0;
  ScrollBox1.HorzScrollBar.Position := 0;
  for i:=0 to ListKrit.Count-1 do begin
    pData := TDemiPanel(ListKrit[i]).Data;
    if pData <> nil then begin
      pData.IndepRect := TDemiPanel(ListKrit[i]).BoundsRect;
    end;
    HitungIndep(ListRect, ParentElement);
  end;
  FUpdateData := False;
  FormPanduan.Close;}
end;

{procedure TFormInterDependency.GetListParent(Element: TTreeElement;
var Stop: Boolean);
begin
  if Element.FirstChild<>nil then begin
    ListParent.Add(Element);
    ComboBox2.Items.Add(Element.DataName);
  end;
end;
}
procedure TFormInterDependency.ComboBox2Change(Sender: TObject);
var vAction: TCloseAction;
begin
{  if Assigned(PrepareObject) then begin
    FormClose(nil, vAction);
    TreeObject.UpdatePosition;
    TreeObject.Invalidate;
    TreeObject.SelectedElement := ListParent[ComboBox2.ItemIndex];
    PrepareObject(nil);
    FormShow(nil);
  end;}
end;

procedure TFormInterDependency.Button2Click(Sender: TObject);
begin
  Close;
end;
```

```
procedure TFormInterDependency.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
begin
  CanClose := Button2.Enabled;
end;

procedure TFormInterDependency.BtnPrevClick(Sender: TObject);
begin
  {if ComboBox2.ItemIndex>0 then begin
    ComboBox2.ItemIndex := ComboBox2.ItemIndex-1;
    ComboBox2Change(nil);
  end;}
end;

procedure TFormInterDependency.BtnNextClick(Sender: TObject);
begin
  {if ComboBox2.ItemIndex<ComboBox2.Items.Count-1 then begin
    ComboBox2.ItemIndex := ComboBox2.ItemIndex+1;
    ComboBox2Change(nil);
  end;
  if ComboBox2.ItemIndex=ComboBox2.Items.Count-1 then begin
    Button2.Enabled := True;
    ComboBox2.Enabled := True;
  end;}
end;

procedure FindUnion(vListRect: TList; vCurrent: Integer;
vListTarget: TStrings);
  function FindItem(vx: string): integer;
  var i: Integer;
    vKetemu: Boolean;
  begin
    result := -1;
    vKetemu := False;
    for i:=0 to vListTarget.Count-1 do begin
      if Pos(vx, vListTarget[i])>0 then begin
        vKetemu := True;
        result := i;
        exit;
      end;
    end;
  end;
var vTarget: Integer;
  pDatai, pDataj: pTDataRect;
  vX: string;
  vidx1, vidx2: Integer;
  vRect: TRect;
  i, j: Integer;
  vIndep: Boolean;
begin
  for i:=0 to vListRect.Count-1 do begin
    pDatai := vListRect[i];
    if FindItem(pDatai^.id)<0 then begin
      vListTarget.Add(pDatai^.id);
    end;
    for j:=i+1 to vListRect.Count-1 do begin
```

```
      pDataj := vListRect[j];
      IntersectRect(vRect, pDatai^.RectData, pDataj^.RectData);
      if not((vRect.Left=0) and (vRect.Top=0) and (vRect.Right=0)
and
        (vRect.Bottom=0)) then begin
        vidx1 := FindItem(pDatai^.id);
        vidx2 := FindItem(pDataj^.id);
        if (vIdx1>=0) and (vIdx2>=0) then begin
          if vIdx1<>vIdx2 then begin
            vListTarget[vIdx1] :=
vListTarget[vIdx1]+vListTarget[vIdx2];
            vListTarget.Delete(vIdx2);
          end;
        end else if vIdx1>=0 then begin
          vListTarget[vidx1] := vListTarget[vidx1]+pDataj^.id;
        end else if vIdx2>=0 then begin
          vListTarget[vidx2] := pDatai^.id+vListTarget[vidx2];
        end;
      end;
    end;
  end;
  {while vCurrent<=vListRect.Count-1 do begin
    vTarget := vCurrent+1;
    pDatai := vListRect[vCurrent];
    vX := pDatai^.id;
    vIdx := vListTarget.Add(vX);
    while (vTarget<=vListRect.Count-1) do begin
      pDataj := vListRect[vTarget];
      IntersectRect(vRect, pDatai^.RectData, pDataj^.RectData);
      if (vRect.Left=0) and (vRect.Top=0) and (vRect.Right=0) and
        (vRect.Bottom=0) then begin
        //FindUnion(vListRect, vTarget, vListTarget);
        //vLIstRect.Delete(vTarget);
        inc(vTarget);
      end else begin
        vX := vX+pDataj^.id;
        vLIstRect.Delete(vTarget);
        //inc(vTarget);
      end;
    end;
    vListTarget[vIdx] := vX;
    inc(vCurrent);
  end;}
end;
{
  vCurrent := 0;
  vTarget := 1;
  vX := '';
  if ListRect.COunt = 1 then begin
    pDatai := ListRect[vCurrent];
    FIndependentList.Add(pDatai^.id;
  end else begin
    while (vTarget<=ListRect.Count-1) do begin
      pDatai := ListRect[vCurrent];
      pDataj := ListRect[vTarget];
      if vX='' then begin
```

```
        vX := vX+pData^.id;
      end;
      vX := pDataj^.id;
      IntersectRect(vRect, pDatai^.RectData, pDataj^.RectData);
      if not ((vRect.Left=0) and (vRect.Top=0) and (vRect.Right=0)
and
        (vRect.Bottom=0)) then begin

      end;
    end;
  end;
}

procedure AddUnionValue(vList: TList; pData: pTDataRect);
var vpData: pTDataRect;
  vx: string;
  i: Integer;
  xx: Extended;
begin
  if Length(pData^.id)>0 then begin
    vx := pData^.id[1];
    for i:=0 to vList.Count-1 do begin
      vpData := vList[i];
      if Pos(vx, vpData^.id)>=1 then begin
        if Length(pData^.id) mod 2 = 0 then begin
          xx := vpData^.Luas-pData^.Luas;
          vpData^.Luas := xx;
          //vpData^.Luas := vpData^.Luas-pData^.Luas;
        end else begin
          xx := vpData^.Luas+pData^.Luas;
          vpData^.Luas := xx;
          //vpData^.Luas := vpData^.Luas+pData^.Luas;
        end;
      end;
    end;
  end;
end;

procedure TFormInterDependency.HitungIndep(ListRect: TList; Parent:
TObject);
var i, j, k: Integer;
  //pElement: TTreeElement;
  pData: pTDataRect;
  pDatai: pTDataRect;
  pDataj: pTDataRect;
  ketemu: boolean;
  vId: String;
  vctr: char;
  vRect: TRect;
  jmSgl: Integer;
  vStart: char;
  vNomor: Integer;
  vJumlahLuas: Extended;
  vPanel: TDemiPanel;
  vLuas: Integer;
  vCurrent, vTarget: Integer;
```

```
  vX: string;
  xListRect: TList;
  vField: TKFieldObject;
  vPos: Integer;
  vJml: Extended;
begin
  ClearListRect(ListRect);
  vStart := 'A';
  vctr := vStart;
  for i:=0 to panelOwner.ComponentCount-1 do begin
    vPanel := TDemiPanel(panelOwner.Components[i]);
    New(pData);
    pData^.Id := vctr;
    pData^.RectData := vPanel.BoundsRect;
    pData^.Luas := 0;
    pData^.BobotLuas := 0;
    ListRect.Add(pData);
    vPanel.RectId := vctr;
    inc(vctr);
  end;
  FIndependentList.Clear;
  xListRect := TList.Create;
  try
    for i:=0 to ListRect.Count-1 do begin
      xListRect.Add(ListRect[i]);
    end;
    if xListRect.Count=1 then begin
      pDatai := xListRect[0];
      FIndependentList.Add(pDatai^.id);
    end else if xListRect.Count>1 then begin
      vCurrent := 0;
      FindUnion(xListRect, vCurrent, FIndependentList);
    end;
  finally
    xListRect.Free;
  end;
  {pElement := Parent.FirstChild;
  while pElement<>nil do begin
    if not ((pElement.IndepRect.Left = 0) and
(pElement.IndepRect.Top = 0) and
      (pElement.IndepRect.Right = 0) and (pElement.IndepRect.Bottom
= 0)) then begin
      New(pData);
      pData^.Id := vctr;
      pData^.RectData := pElement.IndepRect;
      ListRect.Add(pData);
      pElement.RectId := vctr;
      inc(vctr);
    end;
    pElement := pElement.NextSibling;
  end;}
  jmSgl := ListRect.Count;
  for i:=0 to ListRect.Count-1 do begin
    pDatai := ListRect[i];
    for j:=i+1 to ListRect.Count-1 do begin
      pDataj := ListRect[j];
```

```
      if (not CekId(pDatai^.id, pDataj^.id)) and
        (not CekId(pDataj^.id, pDatai^.id)) then begin
        IntersectRect(vRect, pDatai^.RectData, pDataj^.RectData);
        if not ((vRect.Left=0) and (vRect.Top=0) and (vRect.Right=0)
and
          (vRect.Bottom=0)) then begin
          k:=0;
          Ketemu := False;
          vid := pDatai^.id+pDataj^.id;
          while k<=ListRect.Count-1 do begin
            pData := ListRect[k];
            if CekId(pData^.id, vid) and CekId(vId, pData^.Id) then
begin
              ketemu := True;
              k := ListRect.Count;
            end;
            k:=k+1;
          end;
          if not Ketemu then begin
            New(pData);
            pData^.id := vId;
            pData^.RectData := vRect;
            pData^.Luas := 0;
            pData^.BobotLuas := 0;
            ListRect.Add(pData);
          end;
        end;
      end;
    end;
  end;
  (* Hitung Luas *)
  for i:=0 to ListRect.Count-1 do begin
    pData := ListRect[i];
    pData^.Luas := (pData^.RectData.Right-pData^.RectData.Left) *
      (pData^.RectData.Bottom-pData^.RectData.Top);
  end;
  {for i:=0 to JmSgl-1 do begin
    pDatai := ListRect[i];
    for j:=JmSgl to ListRect.Count-1 do begin
      pDataj := ListRect[j];
      if CekId(pDatai^.Id, pDataJ^.Id) then begin
        if ((Length(pDataj^.Id) div 2)*2) = Length(pDataj^.Id) then
begin
          pDatai^.Luas := pDatai^.Luas-pDataj^.Luas;
        end else begin
          pDatai^.Luas := pDatai^.Luas+pDataj^.Luas;
        end;
      end;
    end;
  end;}
  vJumlahLuas := 0;
  for i:=0 to JmSgl-1 do begin
    pDatai := ListRect[i];
    with pDatai^.RectData do begin
      vLuas := (Right-Left)*(Bottom-Top);
    end;
```

```
    vJumlahLuas := vJumlahLuas+vLuas;
  end;
  FJumlahLuas := vJumlahLuas;
  {for i:=0 to JmSgl-1 do begin
    pDatai := ListRect[i];
    if Length(pDatai.id) mod 2 = 0 then begin
      FJumlahLuas := FJumlahLuas-pDatai^.Luas;
    end else begin
      FJumlahLuas := FJumlahLuas+pDatai^.Luas;
    end;
  end;}
  {for i:=0 to JmSgl-1 do begin
    pDatai := ListRect[i];
    if vJumlahLuas = 0 then begin
      pDatai^.BobotLuas := 0;
    end else begin
      pDatai^.BobotLuas := pDatai^.Luas/vJumlahLuas;
    end;
  end;}
  for i:=0 to ListRect.Count-1 do begin
    pDatai := ListRect[i];
    if vJumlahLuas = 0 then begin
      pDatai^.BobotLuas := 0;
    end else begin
      pDatai^.BobotLuas := pDatai^.Luas/vJumlahLuas;
    end;
  end;
  for i:=0 to PanelOwner.ComponentCount-1 do begin
    vPanel := TDemiPanel(PanelOwner.Components[i]);
    vnomor := Ord(vPanel.RectId)-Ord(vStart);
    pData := ListRect[vNomor];
    {if pElement.PreferensiLokal <> 0 then begin
      //pElement.IndepValue :=
pData^.BobotLuas/pElement.PreferensiLokal;
    end else begin
      //pElement.IndepValue := 0;
    end;}
  end;
  for i:=0 to FListSegment.Count-1 do begin
    pData := FListSegment[i];
    Dispose(pData);
  end;
  FListSegment.Clear;
  for i:=0 to FIndependentList.Count-1 do begin
    New(pData);
    pData^.id := FIndependentList[i];
    pData^.Luas := 0;
    pData^.BobotLuas := 0;
    FListSegment.Add(pData);
  end;
  for i:=0 to ListRect.Count-1 do begin
    pData := ListRect[i];
    AddUnionValue(FListSegment, pData);
  end;
  FJumlahLuasIndep := 0;
  for i:=0 to FListSegment.Count-1 do begin
```

```
    pData := FListSegment[i];
    FJumlahLuasIndep := FJumlahLuasIndep+pData^.Luas;
  end;
  for i:=0 to FListSegment.Count-1 do begin
    pDatai := FListSegment[i];
    if Length(pDatai^.id)>1 then begin
      vJml := 0;
      for j:=1 to Length(pDatai^.id) do begin
        try
        vPos := Ord(pDatai^.id[j])-65;
        vField := FCritParent[vPos];
        vField :=
TKBlockField(vFIeld).BlockData.FieldByName('Criteria');
        vJml := vJml+TKBlockField(vFIeld).BlockData.TempReal2;
        except
        end;
      end;
      pDatai^.JmlBobotAwal := vJml;
    end;
  end;
  for i:=0 to jmSgl-1 do begin
    pDatai := ListRect[i];
    for j:=0 to FListSegment.Count-1 do begin
      pDataj := FListSegment[j];
      if Pos(pDatai^.id, pDataj^.id)>0 then begin
        vFIeld := FCritParent[i];
        vField :=
TKBlockField(vFIeld).BlockData.FieldByName('Criteria');
        if Length(pDataj^.id)<=1 then begin
          TKBlockField(vFIeld).BlockData.TempReal4 :=
{TKBlockField(vFIeld).BlockData.TempReal2*}
            pDataj^.Luas/FJumlahLuasIndep;
          TKBlockField(vFIeld).BlockData.TempReal3 :=
TKBlockField(vFIeld).BlockData.TempReal4*
            FCritParent.TempReal3;
          break;
        end else begin
          try
            if pDataj^.JmlBobotAwal>0 then begin
              TKBlockField(vFIeld).BlockData.TempReal4 :=
(pDatai^.Luas/
                (FJumlahLuasIndep*pDataj^.JmlBobotAwal))*
                (pDataj^.Luas/FJumlahLuasIndep);
            end;
            TKBlockField(vFIeld).BlockData.TempReal3 :=
TKBlockField(vFIeld).BlockData.TempReal4*
              FCritParent.TempReal3;
          except
          end;
          break;
        end;
      end;
    end;
  end;
  if FIndependentList.Count=0 then begin
    StringGrid2.RowCount := 2;
```

```
    StringGrid2.Rows[0].Text := '';
  end else begin
    StringGrid2.RowCount := FIndependentList.Count+1;
    {for i:=0 to FIndependentList.Count-1 do begin
      StringGrid2.Cells[0, i+1] := FIndependentList[i];
    end;}
  end;
  DrawGrid1.RowCount := ListRect.Count+1;
  {pElement := Parent.FirstChild;
  while pElement<>nil do begin
    vnomor := Ord(pElement.RectId)-Ord(vStart);
    pData := ListRect[vNomor];
    if pElement.PreferensiLokal <> 0 then begin
    pElement.IndepValue :=
pData^.BobotLuas/pElement.PreferensiLokal;
    end else begin
      pElement.IndepValue := 0;
    end;
    pElement := pElement.NextSibling;
  end;}
end;

function CekId(id1, id2: string): Boolean;
var i: integer;
  v: char;
begin
  i:=1;
  result := true;
  while i<=Length(id1) do begin
    v := id1[i];
    if pos(v, id2)<=0 then begin
      result := False;
      i:=Length(id1);
    end;
    inc(i);
  end;
end;

procedure ClearListRect(ListRect: TList);
var i: Integer;
  pData: pTDataRect;
begin
  for i:=0 to ListRect.Count-1 do begin
    pData := ListRect[i];
    Dispose(pData);
  end;
  ListRect.Clear;
end;


procedure TFormInterDependency.SetCritParent(const Value:
TKBlockObject);
var i: Integer;
  vRect: TRect;
  vx, vy: Integer;
  vPanel: TDemiPanel;
```

```
  vField: TKFieldObject;
  xField: TKFieldObject;
  vcx, vcy: Integer;
  vSt: Integer;
  vwidth, vheight: Integer;
  vPrefValue: Extended;
  vRatio: Extended;
  a: Integer;
begin
  for i:=PanelOwner.ComponentCount-1 downto 0 do begin
    PanelOwner.Components[i].Free;
  end;
  FCritParent := Value;
  Edit1.Text := '';
  try
    if FCritParent.OwnerField.Owner.OwnerField<>nil then begin
      Edit1.Text :=
FCritParent.OwnerField.Owner.OwnerField.FieldName;
    end else begin
      Edit1.Text := 'Goal';
    end;
  except
  end;
  for i:=0 to Value.FieldCount-1 do begin
    xField := Value[i];
    vcx := xField.TagValue1;
    vcy := xField.TagValue2;
    vSt := xField.TagValue3;
    vField :=
TKBlockField(xField).BlockData.FieldByName('Criteria');
    vPrefValue := TKBlockField(vField).BlockData.TempReal2;
    a := Panel1.Height;
    vHeight := Round(HitungAkar(vPrefValue*a*a, 2));
    vWidth := Round(3*HitungAkar(vPrefValue*a*a, 2));

    {vRatio := HitungAkar(vPrefValue, 2);
    vWidth := Round(Panel1.Width * vRatio);
    vHeight := Round(Panel1.Height * vRatio);}
    {vWidth := Round(Panel1.Width * vPrefValue);
    vHeight := Round(Panel1.Height * vPrefValue);}
    if vSt = 0 then begin
      vx := Panel1.Left;
      vy := i*(Panel1.Top+Panel1.Height)+Panel1.Top;
      vcx := vx;
      vcy := vy;
    end else begin
      vx := vcx;
      vy := vcy;
    end;
    xField.TagValue1 := vcx;
    xField.TagValue2 := vcy;
    vRect := Rect(vx, vy, vx+vWidth, vy+vHeight);
    vPanel := TDemiPanel.Create(PanelOwner);
    vPanel.Tag := i;
    vPanel.BoundsRect := vRect;
    vPanel.Parent := ScrollBox1;
```

```
    vPanel.Color := MyBackColor[i+1];
    vPanel.OnMouseDown := Panel1MouseDown;
    vPanel.OnDraw := MyDrawPanel;
  end;
  HitungIndep(ListRect, nil);
  StringGrid1.ColCount := 3;
  StringGrid2.ColCount := 3;
  DrawGrid1.ColCount := 2;
  StringGrid1.RowCount := FCritParent.FieldCount+1;
  StringGrid1.Refresh;
  StringGrid2.Refresh;
  DrawGrid1.Refresh;
end;

procedure TFormInterDependency.DrawGrid1DrawCell(Sender: TObject;
ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
  procedure DrawIrisan(Canvas: TCanvas; vTeks: string; xRect:
TRect);
  var vLeft: Integer;
    vFontName: string;
    i: Integer;
  begin
    vLeft := xRect.Left+1;
    vFontName := Canvas.Font.Name;
    for i:=1 to Length(vTeks) do begin
      if i>1 then begin
        Canvas.Font.Name := 'Symbol';
        Canvas.TextOut(vLeft, xRect.Top+1, chr(199));
        vLeft := vLeft+Canvas.TextWidth(chr(199))+2;
      end;
      Canvas.Font.Name := vFontName;
      Canvas.TextOut(vLeft, xRect.Top+1, vTeks[i]);
      vLeft := vLeft+Canvas.TextWidth(vTeks[i])+2;
    end;
  end;
var vTeks: string;
  pData: pTDataRect;
  vLuas: Integer;
begin
  vTeks := '';
  if ARow=0 then begin
    case ACol of
      0:vTeks := 'Area';
      1: vTeks := 'Size';
    end;
  end else begin
    pData := ListRect.Items[ARow-1];
    with pData^.RectData do begin
      vLuas := (Right-Left)*(Bottom-Top);
    end;
    case ACol of
      0: vTeks := pData^.Id;
      //1: vTeks := FormatFloat('0.####', vLuas);
      1: vTeks := FormatFloat('0.####', vLuas/FJumlahLuas);
    end;
```

```
  end;
  if (ACol = 0) and (ARow>0) then begin
    DrawIrisan(DrawGrid1.Canvas, vTeks, Rect);
  end else begin
    DrawGrid1.Canvas.TextOut(Rect.Left+1, Rect.Top+1, vTeks);
  end;
end;

procedure TFormInterDependency.StringGrid2DrawCell(Sender: TObject;
ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
  procedure DrawUnion(Canvas: TCanvas; vTeks: string; xRect: TRect);
  var vLeft: Integer;
    vFontName: string;
    i: Integer;
  begin
    vLeft := xRect.Left+1;
    vFontName := Canvas.Font.Name;
    for i:=1 to Length(vTeks) do begin
      if i>1 then begin
        Canvas.Font.Name := 'Symbol';
        Canvas.TextOut(vLeft, xRect.Top+1, chr(200));
        vLeft := vLeft+Canvas.TextWidth(chr(200))+2;
      end;
      Canvas.Font.Name := vFontName;
      Canvas.TextOut(vLeft, xRect.Top+1, vTeks[i]);
      vLeft := vLeft+Canvas.TextWidth(vTeks[i])+2;
    end;
  end;
var pData: pTDataRect;
  vTeks: string;
begin
  //
  try
    vTeks := '';
    if ARow=0 then begin
      if ACol=0 then begin
        vTeks := 'Area';
      end else if ACol=1 then begin
        vTeks := 'Size';
      end else if ACol = 2 then begin
        vTeks := 'Weight';
      end;
    end else begin
      pData := FListSegment[ARow-1];
      if ACol=0 then begin
        vTeks := pData^.id;
      end else if ACol=1 then begin
        vTeks := FormatFloat('0.####', pData^.Luas/FJumlahLuas);
      end else if ACol = 2 then begin
        vTeks := FormatFloat('0.####',
pData^.Luas/FJumlahLuasIndep);
      end;
    end;
    if (ACol = 0) and (ARow>0) then begin
      DrawUnion(StringGrid2.Canvas, vTeks, Rect);
```

```
    end else begin
      StringGrid2.Canvas.TextOut(Rect.Left+1, Rect.Top+1, vTeks);
    end;
  except
  end;
end;

end.
```

```
unit Fpanduan;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, Grids, StdCtrls, Buttons, ExtCtrls;

type
  TFormPanduan = class(TForm)
    Panel1: TPanel;
    StringGrid2: TStringGrid;
    Panel2: TPanel;
    BitBtn1: TBitBtn;
    Label1: TLabel;
    CheckBox1: TCheckBox;
    procedure BitBtn1Click(Sender: TObject);
    procedure CheckBox1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormPanduan: TFormPanduan;

implementation

{$R *.DFM}

procedure TFormPanduan.BitBtn1Click(Sender: TObject);
begin
  Close;
end;

procedure TFormPanduan.CheckBox1Click(Sender: TObject);
begin
  if CheckBox1.Checked then begin
    FormStyle := fsStayOnTop;
  end else begin
    FormStyle := fsNormal;
  end;
end;

end.
```

```
unit SPNL;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, ExtCtrls, Menus;

type
  TOnDrawEvent = procedure(Sender: TObject; Canvas: TCanvas;
ObjBound: TRect) of object;
  TDemiPanel = Class (TCustomPanel)
  private
    FTopColor, FBottomColor: TColor;
    FBackPicture: TPicture;
    FDrawToDC: HDC;
    FDrawHeight, FDrawWidth: Integer;
    FOnDraw: TOnDrawEvent;
    FRectId: char;
    procedure SetPicture(Value: TPicture);
    procedure PictureChanged(Sender: TObject);
    procedure SetRectId(const Value: char);
  protected
    procedure Paint; override;
    procedure WMEreaseBkgnd(var Message: TMessage); message
WM_ERASEBKGND;
  public
    Data: Pointer;
    AbsBounds: TRect;
    property OnDraw: TOnDrawEvent read FOnDraw write FOnDraw;
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
  published
    property RectId: char read FRectId write SetRectId;
    property TopColor: TColor read fTopColor write fTopColor;
    property BottomColor: TColor read fBottomColor write
fBottomColor;
    property BackPicture: TPicture read FBackPicture write
SetPicture;
    property Align;
    property Alignment;
    property BevelInner;
    property BevelOuter;
    property BevelWidth;
    property BorderWidth;
    property BorderStyle;
    property DragCursor;
    property DragMode;
    property Enabled;
    property Caption;
    property Color;
    property Ctl3D;
    property Font;
    property Locked;
    property ParentColor;
```

```
    property ParentCtl3D;
    property ParentFont;
    property ParentShowHint;
    property PopupMenu;
    property ShowHint;
    property TabOrder;
    property TabStop;
    property Visible;
    property OnClick;
    property OnDblClick;
    property OnDragDrop;
    property OnDragOver;
    property OnEndDrag;
    property OnEnter;
    property OnExit;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
    property OnResize;
  end;

procedure Register;

implementation

constructor TDemiPanel.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  Parent := AOwner as TWinControl;
  fBackPicture := TPicture.Create;
  FBackPicture.OnChange := PictureChanged;
  fTopColor := clBtnHighLight;
  fBottomColor := clBtnShadow;
end;

destructor TDemiPanel.Destroy;
begin
  fBackPicture.Destroy;
  inherited Destroy;
end;

procedure TDemiPanel.Paint;
var
  xRect: TRect;
  TopColor, BottomColor: TColor;
  Text: array[0..255] of Char;
  FontHeight: Integer;
  Row, Column: integer;
  xl, xt, xw, xh: integer;
  vCanvas: TControlCanvas;
const
  Alignments: array[TAlignment] of Word = (DT_LEFT, DT_RIGHT,
DT_CENTER);
```

```
  procedure AdjustColors(Bevel: TPanelBevel);
  begin
    TopColor := fTopColor;
    if Bevel = bvLowered then TopColor := fBottomColor;
    BottomColor := fBottomColor;
    if Bevel = bvLowered then BottomColor := fTopColor;
  end;

begin
  xRect := GetClientRect;
  if BevelOuter <> bvNone then begin
    AdjustColors(BevelOuter);
    Frame3D(Canvas, xRect, TopColor, BottomColor, BevelWidth);
  end;
  Frame3D(Canvas, xRect, Color, Color, BorderWidth);
  if BevelInner <> bvNone then begin
    AdjustColors(BevelInner);
    Frame3D(Canvas, xRect, TopColor, BottomColor, BevelWidth);
  end;
  if Assigned(FOnDraw) then begin
    vCanvas := TControlCanvas.Create;
    vCanvas.Control := Self;
    FOnDraw(Self, vCanvas, Rect(Left, Top, Left+Width, Top+Height));
    vCanvas.Destroy;
  end else begin
    inherited Paint;
  end;
(*  with Canvas do begin
    {Drawing BackGround Image}
    if (BackPicture.Bitmap.Width <> 0) and
(BackPicture.Bitmap.Height<> 0) then begin
      FDrawHeight := Rect.Bottom - Rect.Top;
      FDrawWidth := Rect.Right - Rect.Left;
{      FDrawToDC := GetWindowDC(Self.Handle);}
      for Row := 0 to FDrawHeight div fBackPicture.Bitmap.Height do
begin
        for Column := 0 to FDrawWidth div fBackPicture.Bitmap.Width
do begin
          xl := Column*fBackPicture.Bitmap.Width+Rect.Left;
          xt := Row*fBackPicture.Bitmap.Height+Rect.Top;
          xw := fBackPicture.Bitmap.Width;
          if (FDrawWidth - xl+Rect.Left) < xw then xw := (FDrawWidth
- xl+Rect.Top);
          xh := fBackPicture.Bitmap.Height;
          if (FDrawHeight - xt+Rect.Top) < xh then xh :=
(FDrawHeight - xt+Rect.Top);
          BitBlt(Handle, xl, xt, xw, xh,
            fBackPicture.Bitmap.Canvas.Handle, 0, 0, SRCCOPY);
        end;
      end;
{      ReleaseDC(Self.Handle, FDrawToDC);}
    end;
    Brush.Color := Color;
    {FillRect(Rect);}
    Brush.Style := bsClear;
    Font := Self.Font;
```

```
    FontHeight := TextHeight('W');
    with Rect do begin
      Top := ((Bottom + Top) - FontHeight) shr 1;
      Bottom := Top + FontHeight;
    end;
    StrPCopy(Text, Caption);
    DrawText(Handle, Text, StrLen(Text), Rect, (DT_EXPANDTABS or
      DT_VCENTER) or Alignments[inherited Alignment]);
  end;
  *)
end;

procedure TDemiPanel.SetPicture(Value: TPicture);
begin
  FBackPicture.Assign(Value);
end;

procedure TDemiPanel.WMEreaseBkgnd(var Message: TMessage);
begin
  {Message.Result := 0;}
  inherited;
end;

procedure TDemiPanel.PictureChanged(Sender: TObject);
begin
{  if AutoSize and (Picture.Width > 0) and (Picture.Height > 0) then
    SetBounds(Left, Top, Picture.Width, Picture.Height);
  if (Picture.Graphic is TBitmap) and (Picture.Width = Width) and
    (Picture.Height = Height) then
    ControlStyle := ControlStyle + [csOpaque] else
    ControlStyle := ControlStyle - [csOpaque];}
  Invalidate;
end;

procedure Register;
begin
  RegisterComponents('KPC', [TDemiPanel]);
end;

procedure TDemiPanel.SetRectId(const Value: char);
begin
  FRectId := Value;
end;

end.
```

```
unit Fsens;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, StdCtrls, {DTree, }ExtCtrls, KFieldObjects,
UDataStru, FIndep;
const
  FormatData = '0.####';
type

  TItemKriteria = record
    NmKriteria: TLabel;
    ScKriteria: TScrollBar;
    LcKriteria: TEdit;
    GlbKriteria: TEdit;
    Block: TKBlockObject;
  end;
  pTItemKriteria = ^TItemKriteria;

  TItemAlt = record
    NmAlt: TLabel;
    NilaiAlt: TEdit;
    Field: TKFieldObject;
  end;
  pTItemAlt = ^TItemAlt;

  TFormSensitivitas = class(TForm)
    Edit4: TEdit;
    Panel1: TPanel;
    Label1: TLabel;
    ComboBox1: TComboBox;
    ScrollBox1: TScrollBox;
    Label2: TLabel;
    Label5: TLabel;
    ScrollBar1: TScrollBar;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Panel3: TPanel;
    Label3: TLabel;
    Label4: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Panel4: TPanel;
    Panel5: TPanel;
    BtnRefresh: TButton;
    Button2: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure ScrollBar1Change(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure BtnRefreshClick(Sender: TObject);
```

```
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
    FListKriteria: TList;
    FListAlt: TList;
    FDataStru: TRDataStru;
    FRefDataStru: TRDataStru;
    FSelected: TKBlockObject;
    FCalcFromProgram: Boolean;
    procedure AddKriteriaObj(vBlock: TKBLockObject);
    procedure CalculatingData;
    procedure BuildDisplay;
    procedure ClearList;
    procedure AddAlternativeObj(vField: TKFieldObject);
    procedure LoadPosition(vNo: Integer);
    procedure PostProcess(vCrit: TKBlockObject; vLevel: Integer);
  public
    { Public declarations }
    RefreshCriteria: TNotifyEvent;
    property Selected: TKBlockObject read FSelected write FSelected;
    procedure SetCritParent(vDataStru: TRDataStru);
    procedure SetParentByName(vName: string);
  end;

var
  FormSensitivitas: TFormSensitivitas;

implementation

{$R *.DFM}

procedure TFormSensitivitas.AddAlternativeObj(vFIeld:
TKFieldObject);
var pIAlt: pTItemAlt;
  vJml: Integer;
  vTinggi: Integer;
begin
  vJml := FListAlt.Count;
  vTinggi := Edit3.Height+5;
  New(pIAlt);
  pIAlt^.NmAlt := TLabel.Create(Self);
  pIAlt^.NmAlt.Parent := Label5.Parent;
  pIAlt^.NmAlt.Left := Label5.Left;
  pIAlt^.NmAlt.Top := (vJml*vTinggi)+Label5.Top;
  pIAlt^.NmAlt.Caption := vFIeld.FieldName;
  pIAlt^.NilaiAlt := TEdit.Create(Self);
  pIAlt^.NilaiAlt.Parent := Label5.Parent;
  pIAlt^.NilaiAlt.Left := Edit3.Left;
  pIAlt^.NilaiAlt.Width := Edit3.Width;
  pIAlt^.NilaiAlt.Top := (vJml*vTinggi)+Edit3.Top;
  pIAlt^.NilaiAlt.ReadOnly := True;
  pIAlt^.FIeld := vField;
  FListAlt.Add(pIAlt);
end;
```

```
procedure TFormSensitivitas.AddKriteriaObj(vBlock: TKBLockObject);
var
  pIKrit: pTItemKriteria;
  vJml: Integer;
  vTinggi: Integer;
  vName: string;
begin
  vTinggi := Edit1.Height+5;
  vJml := FListKriteria.Count;
  New(pIKrit);
  vName := vBlock.OwnerField.Owner.OwnerField.FieldName;
  pIKrit^.NmKriteria := TLabel.Create(Self);
  pIKrit^.NmKriteria.Parent := Label2.Parent;
  pIKrit^.NmKriteria.Left := Label2.Left;
  pIKrit^.NmKriteria.Top := (vJml*vTinggi)+Label2.Top;
  pIKrit^.NmKriteria.Caption := vName;
  pIKrit^.ScKriteria := TScrollBar.Create(Self);
  pIKrit^.ScKriteria.Parent := Label2.Parent;
  pIKrit^.ScKriteria.Left := ScrollBar1.Left;
  pIKrit^.ScKriteria.Top := (vJml*vTinggi)+ScrollBar1.Top;
  pIKrit^.ScKriteria.Width := ScrollBar1.Width;
  pIKrit^.ScKriteria.Height := ScrollBar1.Height;
  pIKrit^.ScKriteria.Max:= 100;
  pIKrit^.ScKriteria.LargeChange := ScrollBar1.LargeChange;
  pIKrit^.ScKriteria.Tag := vJml;
  pIKrit^.ScKriteria.OnChange := ScrollBar1Change;
  pIKrit^.LcKriteria := TEdit.Create(Self);
  pIKrit^.LcKriteria.Parent := Label2.Parent;
  pIKrit^.LcKriteria.Left := Edit1.Left;
  pIKrit^.LcKriteria.Top := (vJml*vTinggi)+Edit1.Top;
  pIKrit^.LcKriteria.Width := Edit1.Width;
  pIKrit^.LcKriteria.ReadOnly := True;
  pIKrit^.GlbKriteria := TEdit.Create(Self);
  pIKrit^.GlbKriteria.Parent := Label2.Parent;
  pIKrit^.GlbKriteria.Left := Edit2.Left;
  pIKrit^.GlbKriteria.Top := (vJml*vTinggi)+Edit2.Top;
  pIKrit^.GlbKriteria.Width := Edit2.Width;
  pIKrit^.GlbKriteria.ReadOnly := True;
  pIKrit^.Block := vBlock;
  FListKriteria.Add(pIKrit);
end;

procedure TFormSensitivitas.CalculatingData;
var vCrit: TKBlockObject;
  xCrit, vAlt: TKBlockObject;
  i, j: Integer;
  var vJml: Extended;
begin
  FDataStru.CalcAllCriteria;
  vCrit := FDataStru.Root;
  xCrit := nil;
  while xCrit <> vCrit do begin
    xCrit := vCrit;
    FormInterDependency.CritParent := vCrit;
    vCrit := FDataStru.GetNextCriteria(vCrit, False);
  end;
```

```
  for i:=0 to FDataStru.AltList.FieldCount-1 do begin
    try
    vAlt := TKBlockField(FDataStru.AltList[i]).BlockData;
    vAlt.TempReal1 := 0;
    except
    end;
  end;
  FDataStru.HitungAllAlternative;
end;

procedure TFormSensitivitas.FormCreate(Sender: TObject);
begin
  FListKriteria := TList.Create;
  FListAlt := TList.Create;
  FDataStru := TRDataStru.Create;
end;

procedure TFormSensitivitas.FormDestroy(Sender: TObject);
begin
  FListKriteria.Free;
  FListAlt.Free;
  FDataStru.Free;
end;

procedure TFormSensitivitas.ScrollBar1Change(Sender: TObject);
var
  vNo: Longint;
  pIKrit: pTItemKriteria;
  vNilai: Extended;
  i: Integer;
  vField: TKFieldObject;
  vJumlah, vJumlah1: Extended;
  vParentValue: Extended;
begin
  //
  if FCalcFromProgram then exit;
  if Sender is TScrollBar then begin
    vNo := TControl(Sender).Tag;
    if (vNo>=0) and (vNo<FListKriteria.Count) then begin
      vNilai := TScrollBar(Sender).Position/TScrollBar(Sender).Max;
      pIKrit := FListKriteria[vNo];
      if FSelected.OwnerField.Owner.OwnerField = nil then begin
        vparentValue := 1;
      end else begin
        vParentValue := FSelected.TempReal3;
      end;
      {RootElement.HitungPreferensi;}
      {Tree.HitungAlternative;}
      vJumlah := 0;
      vJumlah1 := 0;
      for i:=0 to FSelected.FieldCount-1 do begin
        vField := FSelected[i];
        vField :=
TKBlockField(vField).BlockData.FieldByName('Criteria');
        vJumlah := vJumlah+TKBlockField(vField).BlockData.TempReal4;
        if i=vNo then begin
```

```
          TKBlockField(vField).BlockData.TempReal4 := vNilai;
          vJumlah1 := vJumlah1+vNilai;
        end else begin
          vJumlah1 :=
vJumlah1+TKBlockField(vField).BlockData.TempReal4;
        end;
      end;
      for i:=0 to FSelected.FieldCount-1 do begin
        vField := FSelected[i];
        vField :=
TKBlockField(vField).BlockData.FieldByName('Criteria');
        if vJumlah1 = 0 then begin
          raise Exception.Create('Pembagian dengan Nol');
        end else begin
          TKBlockField(vField).BlockData.TempReal4 :=

TKBlockField(vField).BlockData.TempReal4*vJumlah/vJumlah1;
          TKBlockField(vField).BlockData.TempReal3 :=
            TKBlockField(vField).BlockData.TempReal4*vParentValue;
          FDataStru.TraceCriteria(TKBlockField(vField).BlockData,
nil, postProcess, nil, True)
        end;
      end;
      pIKrit^.GlbKriteria.Text := FormatFloat(FormatData,
vNilai*vParentValue);
      pIKrit^.LcKriteria.Text := FormatFloat(FormatData, vNilai);
      FDataStru.HitungAllAlternative;
      LoadPosition(vNo);
    end;
  end;

end;

procedure TFormSensitivitas.SetCritParent(vDataStru: TRDataStru);
var vStream: TMemoryStream;
  i: Integer;
  vField: TKFieldObject;
  vBlock: TKBlockObject;
  vName: string;
begin
  vStream := TMemoryStream.Create;
  try
    FRefDataStru := vDataStru;
    FRefDataStru.StruData.SaveToStream(vStream);
    vStream.Position := 0;
    FDataStru.StruData.LoadFromStream(vStream);
    FDataStru.SynchronizeData;
    CalculatingData;
    FSelected := FDataStru.Root;
    ComboBox1.Items.Clear;
    for i:=0 to FDataStru.CritList.Count-1 do begin
      vBLock := FDataStru.CritList[i];
      if vBlock.FieldCount>0 then begin
        if vBlock.OwnerField.Owner.OwnerField<>nil then begin
          vName := vBlock.OwnerField.Owner.OwnerField.FieldName;
        end else begin
```

```
          vName := 'Goal';
        end;
        ComboBox1.Items.Add(vName);
        ComboBox1.ItemIndex := 0;
      end;
    end;
    BuildDisplay;
  finally
    vStream.Free;
  end;
end;

procedure TFormSensitivitas.ClearList;
var i: Integer;
  pIKrit: pTItemKriteria;
  pIAlt: pTItemAlt;
begin
  for i:=0 to FListKriteria.Count-1 do begin
    pIKrit := FListKriteria[i];
    pIKrit^.NmKriteria.Free;
    pIKrit^.ScKriteria.Free;
    pIKrit^.LcKriteria.Free;
    pIKrit^.GlbKriteria.Free;
    Dispose(pIKrit);
  end;
  FListKriteria.Clear;
  for i:=0 to FListAlt.Count-1 do begin
    pIAlt := FListAlt[i];
    pIAlt^.NmAlt.Free;
    pIAlt^.NilaiAlt.Free;
    Dispose(PIAlt);
  end;
  FListAlt.Clear;
end;

procedure TFormSensitivitas.BuildDisplay;
var i: Integer;
  vField: TKFieldObject;
  vBLock: TKBlockObject;
begin
  FCalcFromProgram := True;
  try
    if FSelected<>nil then begin
      ClearList;
      for i:=0 to FSelected.FieldCount-1 do begin
        vField := FSelected[i];
        vField :=
TKBlockField(vField).BlockData.FieldByName('Criteria');
        vBlock := TKBlockField(vField).BlockData;
        AddKriteriaObj(vBlock);
      end;
    end;
    for i:=0 to FDataStru.AltList.FieldCount-1 do begin
      AddAlternativeObj(FDataStru.AltList[i]);
    end;
    LoadPosition(-1);
```

```
  finally
    FCalcFromProgram := False;
  end;
end;

procedure TFormSensitivitas.SetParentByName(vName: string);
var idx: Integer;
  vBlock: TKBlockObject;
begin
  idx := FDataStru.FindCriteria(vName);
  vBlock := nil;
  try
    vBlock := FDataStru.CritList[idx];
    if vBlock.FieldCount=0 then begin
      vBlock := nil;
    end;
  except
    vBlock := nil;
  end;
  if vBlock <> nil then begin
    FSelected := vBlock;
    BuildDisplay;
  end;
end;

procedure TFormSensitivitas.ComboBox1Change(Sender: TObject);
var vName: string;
begin
  if FCalcFromProgram then exit;
  if ComboBox1.ItemIndex>=0 then begin
    vName := ComboBox1.Items[ComboBox1.ItemIndex];
    SetParentByName(vName);
    CalculatingData;
  end;
end;

procedure TFormSensitivitas.LoadPosition(vNo: Integer);
var i: Integer;
  pIKrit: pTItemKriteria;
  pIAlt: pTItemAlt;
  vBlock: TKBlockObject;
begin
  {if Assigned(RefreshingAllValue) then begin
    RefreshingAllValue;
  end;}
  FCalcFromProgram := True;
  try
  for i:=0 to FListKriteria.Count-1 do begin
    if i<>vNo then begin
      pIKrit := FListKriteria[i];
      vBlock := pIKrit^.Block;
      pIKrit^.GlbKriteria.Text := FormatFloat(FormatData,
vBlock.TempReal3);
      pIKrit^.LcKriteria.Text := FormatFloat(FormatData,
vBlock.TempReal4);
```

```
      pIKrit^.ScKriteria.Position :=
Round(vBlock.TempReal4*pIKrit^.ScKriteria.Max);
    end;
  end;
  for i:=0 to FListAlt.Count-1 do begin
    pIAlt := FListAlt[i];
    pIAlt^.NilaiAlt.Text := FormatFloat(FormatData,
TKBlockField(pIAlt^.Field).
      BlockData.TempReal1);
  end;
  finally
    FCalcFromProgram :=False;
  end;
end;

procedure TFormSensitivitas.BtnRefreshClick(Sender: TObject);
var vName: string;
begin
  vName := ComboBox1.Items[ComboBox1.ItemIndex];
  SetCritParent(FRefDataStru);
  COmboBox1.ItemIndex := ComboBox1.Items.IndexOf(vName);
  ComboBox1Change(ComboBox1);
  LoadPosition(-1);
end;

procedure TFormSensitivitas.Button2Click(Sender: TObject);
begin
  Close;
end;

procedure TFormSensitivitas.PostProcess(vCrit: TKBlockObject;
  vLevel: Integer);
var vx, vy: Integer;
  vFIeld: TKFieldObject;
  i: Integer;
  vParentValue: Extended;
begin
  if vCrit.OwnerField.Owner.OwnerField = nil then begin
    vParentValue := 1;
  end else begin
    vParentValue :=
vCrit.OwnerField.Owner.OwnerField.Owner.TempReal3;
  end;
  vCrit.TempReal3 := vCrit.TempReal4*vParentValue;
end;

end.
```

```
unit Fgsens;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, ExtCtrls, StdCtrls, {DTree, }TokUtils,
KFieldObjects, UDataStru;

const
  ListType: array[0..4] of TPenStyle =
    (psSolid, psDash, psDot, psDashDot, psDashDotDot);
  ListColor: array[0..8] of TColor =
    (clNavy, clGreen, clPurple, clBlue, clRed, clLime, clFuchsia,
clAqua, clYellow);


type
  TItemAlt = record
    AltImage: TImage;
    AltItem: TKBLockField;
    JenisGaris: Integer;
    Nilai0: Extended;
    Nilai1: Extended;
  end;
  pTItemAlt = ^TItemAlt;
  TFormSensGraph = class(TForm)
    Panel2: TPanel;
    Panel3: TPanel;
    Image1: TImage;
    ScrollBox1: TScrollBox;
    Shape1: TShape;
    LabelFont: TLabel;
    Shape2: TShape;
    ImageHint: TImage;
    VertLine: TShape;
    VirtualShape: TShape;
    Panel6: TPanel;
    Label2: TLabel;
    ComboBox2: TComboBox;
    Panel7: TPanel;
    Label1: TLabel;
    ComboBox1: TComboBox;
    Panel4: TPanel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure ScrollBox1Click(Sender: TObject);
    procedure FormResize(Sender: TObject);
    procedure Image1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Image1MouseMove(Sender: TObject; Shift: TShiftState;
X,
```

```
      Y: Integer);
    procedure Image1MouseUp(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure ComboBox2Change(Sender: TObject);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
    FDataStru: TRDataStru;
    FRefDataStru: TRDataStru;
    FSelected: TKBlockObject;
    FListKriteria: TList;
    FListAlt: TList;
    GraphMargin: TRect;
    SelectNumber: Integer;
    VertLinePos: Extended;
    FWidth: Integer;
    DownState: Boolean;
    FCalcFromProgram: Boolean;
    procedure CalculatingData;
    procedure SetAlternative;
    procedure ClearList;
    procedure RefreshLegend;
    procedure SetPenStyle(Canvas: TCanvas; StyleNo: Integer; Mode:
Byte);
    procedure RefreshGraph;
    procedure ImageClick(Sender: TObject);
    procedure DrawHint;
    procedure ShowCurrentSelect;
    procedure SetVertLine;
    procedure SetParentByName(vName: string);
    procedure RefreshGraphData;
    procedure PostProcess(vCrit: TKBlockObject; vLevel: Integer);
  public
    { Public declarations }
    property Selected: TKBlockObject read FSelected write FSelected;
    procedure SetCritParent(vDataStru: TRDataStru);
  end;

var
  FormSensGraph: TFormSensGraph;

implementation

uses Findep;

{$R *.DFM}

const
  FloatFormat = '0.00';


procedure TFormSensGraph.Button1Click(Sender: TObject);
begin
  Close;
end;
```

```
procedure TFormSensGraph.SetCritParent(vDataStru: TRDataStru);
var vStream: TMemoryStream;
  i: Integer;
  vField: TKFieldObject;
  vBlock: TKBlockObject;
  vName: string;
begin
  vStream := TMemoryStream.Create;
  try
    FRefDataStru := vDataStru;
    FRefDataStru.StruData.SaveToStream(vStream);
    vStream.Position := 0;
    FDataStru.StruData.LoadFromStream(vStream);
    FDataStru.SynchronizeData;
    CalculatingData;
    FSelected := FDataStru.Root;
    ComboBox2.Items.Clear;
    for i:=0 to FDataStru.CritList.Count-1 do begin
      vBLock := FDataStru.CritList[i];
      if vBlock.OwnerField.Owner.OwnerField<>nil then begin
        vName := vBlock.OwnerField.Owner.OwnerField.FieldName;
        ComboBox2.Items.Add(vName);
      end;
    end;
    SetAlternative;
    RefreshLegend;
    RefreshGraph;
  finally
    vStream.Free;
  end;
end;

procedure TFormSensGraph.CalculatingData;
var vCrit: TKBlockObject;
  xCrit, vAlt: TKBlockObject;
  i, j: Integer;
  var vJml: Extended;
begin
  FDataStru.CalcAllCriteria;
  vCrit := FDataStru.Root;
  xCrit := nil;
  while xCrit <> vCrit do begin
    xCrit := vCrit;
    FormInterDependency.CritParent := vCrit;
    vCrit := FDataStru.GetNextCriteria(vCrit, False);
  end;
  for i:=0 to FDataStru.AltList.FieldCount-1 do begin
    try
    vAlt := TKBlockField(FDataStru.AltList[i]).BlockData;
    vAlt.TempReal1 := 0;
    except
    end;
  end;
  FDataStru.HitungAllAlternative;
end;
```

```
procedure TFormSensGraph.FormCreate(Sender: TObject);
begin
  FListKriteria := TList.Create;
  FListAlt := TList.Create;
  FDataStru := TRDataStru.Create;
  ComboBox1.ItemIndex := 0;
  GraphMargin.Left := 50;
  GraphMargin.Top := 40;
  GraphMargin.Right := 50;
  GraphMargin.Bottom := 50;
end;

procedure TFormSensGraph.FormDestroy(Sender: TObject);
begin
  FListKriteria.Free;
  FListAlt.Free;
  FDataStru.Free;
end;

procedure TFormSensGraph.SetAlternative;
var i: Integer;
  pAlt: pTItemAlt;
  //pData: TTreeElement;
begin
  ClearList;
  for i:=0 to FRefDataStru.AltList.FieldCount-1 do begin
    New(pAlt);
    pAlt^.AltImage := TImage.Create(Self);
    pAlt^.AltImage.Picture.Bitmap.Width := Shape1.Width;
    pAlt^.AltImage.Picture.Bitmap.Height := Shape1.Height;
    pAlt^.AltImage.AutoSize := True;
    pAlt^.AltImage.Parent := Shape1.Parent;
    pAlt^.AltImage.Tag := i;
    pAlt^.AltImage.Canvas.Font.Assign(LabelFont.Font);
    pAlt^.AltImage.OnClick := ImageClick;
    pAlt^.AltItem := TKBlockField(FRefDataStru.AltList[i]);
    pAlt^.Nilai0 := 0;
    pAlt^.Nilai1 := 0;
    FListAlt.Add(pAlt);
  end;
end;

procedure TFormSensGraph.ClearList;
var i: Integer;
  pAlt: pTItemAlt;
begin
  for i:=0 to FListAlt.Count-1 do begin
    pAlt := FListAlt[i];
    pAlt^.AltImage.Free;
    Dispose(pAlt);
  end;
  FListAlt.Clear;
end;
```

```
procedure TFormSensGraph.RefreshLegend;
var i: Integer;
  pAlt: pTItemAlt;
  vxLeft, vxTop: Integer;
  vJmlHorz: Integer;
  vWidth, vHeight: Integer;
  dx: Integer;
  vRect: TRect;
  vName: string;
begin
  vJmlHorz := (ScrollBox1.Width-24) div Shape1.Width;
  if vJmlHorz = 0 then vJmlHorz := 1;
  vWidth := Shape1.Width+5;
  vHeight := Shape1.Height+5;
  dx := 5;
  for i:=0 to FListAlt.Count-1 do begin
    pAlt := FListAlt[i];
    pAlt^.JenisGaris := i;
    vxTop := i div vJmlHorz;
    vxLeft := i mod vJmlHorz;
    pAlt^.AltImage.Left := (vxLeft*vWidth)+Shape1.Left;
    pAlt^.AltImage.Top := (vxTop*vHeight)+Shape1.Top;
    SetPenStyle(pAlt^.AltImage.Canvas, -1, 0);
    pAlt^.AltImage.Canvas.Rectangle(0, 0, pAlt^.AltImage.Width,
pAlt^.AltImage.Height);
    SetPenStyle(pAlt^.AltImage.Canvas, pAlt^.JenisGaris,
ComboBox1.ItemIndex);
    pAlt^.AltImage.Canvas.MoveTo(dx, Shape1.Height div 2);
    pAlt^.AltImage.Canvas.LineTo(Shape1.Width div 3, Shape1.Height
div 2);
    vRect := Rect((Shape1.Width div 3)+4, 0, Shape1.Width,
Shape1.Height);
    vName := pAlt^.AltItem.FieldName;
    TextDraw(pAlt^.AltImage.Canvas, vName,
      DT_VCENTER+DT_CENTER+DT_SINGLELINE, vRect);
  end;
end;

procedure TFormSensGraph.SetPenStyle(Canvas: TCanvas; StyleNo:
Integer; Mode: Byte);
var vPenStyle, vColor: Integer;
  vMaxStyle: Integer;
  vStyleNo: Integer;
  vPembagi: Integer;
  LebarGaris: Integer;
begin
  LebarGaris := 1;
  vStyleNo := StyleNo;
  if Mode = 0 then begin
    vPembagi := (High(ListType)-Low(ListType)+1);
  end else begin
    vPembagi := (High(ListColor)-Low(ListColor)+1);
  end;
  vMaxStyle := (High(ListType)-Low(ListType)+1)*(High(ListColor)-
Low(ListColor)+1);
```

```
  if vStyleNo>=vMaxStyle then begin
    Canvas.Pen.Width := (vStyleNo div vMaxStyle)+LebarGaris;
    vStyleNo := vStyleNo mod vMaxStyle;
  end else begin
    Canvas.Pen.Width := LebarGaris;
  end;
  if StyleNo = -1 then begin
    Canvas.Pen.Style := psSolid;
    Canvas.Pen.Color := clBlack;
  end else if StyleNo <= -2 then begin
    Canvas.Pen.Style := psClear;
  end else begin
    if Mode = 0 then begin
      Canvas.Pen.Style := ListType[vStyleNo mod vPembagi];
      Canvas.Pen.Color := ListColor[vStyleNo div vPembagi];
    end else if mode = 1 then begin
      Canvas.Pen.Style := ListType[vStyleNo div vPembagi];
      Canvas.Pen.Color := ListColor[vStyleNo mod vPembagi];
    end else begin
      Canvas.Pen.Style := psSolid;
      Canvas.Pen.Width := Canvas.Pen.Width+1+ vStyleNo div vPembagi;
      Canvas.Pen.Color := ListColor[vStyleNo mod vPembagi];
    end;
  end;
end;

procedure TFormSensGraph.RefreshGraph;
var vRect: TRect;
  GraphRect: TRect;
  vWidth: Integer;
  vHeight: Integer;
  vTeks: string;
  dx, dy: Integer;
  vX, vY: Integer;
  i: Integer;
  pAlt: pTItemAlt;
begin
  dx := 10;
  dy := 10;
  Image1.Picture.Bitmap.Width := 0;
  Image1.Picture.Bitmap.Height := 0;
  Image1.Picture.Bitmap.Width := Image1.Width;
  Image1.Picture.Bitmap.Height := Image1.Height;
  Image1.Canvas.Pen.Width := 1;
  Image1.Canvas.Pen.Color := clBlack;
  Image1.Canvas.Pen.Style := psSolid;
  Image1.Canvas.Rectangle(0, 0, Image1.Width, Image1.Height);
  Image1.Canvas.Pen.Width := 2;
  GraphRect := Rect(GraphMargin.Left, GraphMargin.Top,
    Image1.Width-GraphMargin.Right, Image1.Height-
GraphMargin.Bottom);
  Image1.Canvas.Rectangle(GraphRect.Left-1, GraphRect.Top-1,
GraphRect.Right+2,
    GraphRect.Bottom+2);
  Image1.Canvas.Pen.Width := 1;
  Image1.Canvas.Brush.Color := clAqua;
```

```
  Image1.Canvas.Brush.Style := bsClear;
  Image1.Canvas.Font.Assign(LabelFont.Font);
  Image1.Canvas.Font.Size := 12;
  vTeks := 'Sensitivity Graph for Criteria
'+ComboBox2.Items[ComboBox2.ItemIndex];
  vHeight := Image1.Canvas.TextHeight(vTeks);
  vRect := Rect(GraphRect.Left, GraphRect.Top-vHeight*2,
GraphRect.Right,
    GraphRect.Top);
  TextDraw(Image1.Canvas, vTeks, DT_CENTER+DT_VCENTER+DT_SINGLELINE,
vRect);
  Image1.Canvas.Font.Assign(LabelFont.Font);
  vTeks := '0.00';
  vHeight := Image1.Canvas.TextHeight(vTeks);
  vWidth := Image1.Canvas.TextWidth(vTeks);
  vRect := Rect(GraphRect.Left-(vWidth div 2), GraphRect.Bottom+dy,
GraphRect.Left+vWidth,
    GraphRect.Bottom+dy+vHeight);
  TextDraw(Image1.Canvas, vTeks, DT_CENTER, vRect);
  vTeks := '1.00';
  vHeight := Image1.Canvas.TextHeight(vTeks);
  vWidth := Image1.Canvas.TextWidth(vTeks);
  vRect := Rect(GraphRect.Right-(vWidth div 2), GraphRect.Bottom+dy,
GraphRect.Right+vWidth,
    GraphRect.Bottom+dy+vHeight);
  TextDraw(Image1.Canvas, vTeks, DT_CENTER, vRect);
  for i:=0 to FListAlt.Count-1 do begin
    pAlt := FListAlt[i];
    SetPenStyle(Image1.Canvas, pAlt^.JenisGaris,
ComboBox1.ItemIndex);
    vY := Round(pAlt^.Nilai0*(GraphRect.Bottom-GraphRect.Top));
    Image1.Canvas.MoveTo(GraphRect.Left, GraphRect.Bottom-vY);
    vTeks := FormatFloat(FloatFormat, pAlt^.Nilai0);
    vWidth := Image1.Canvas.TextWidth(vTeks);
    vHeight := Image1.Canvas.TextHeight(vTeks);
    vRect := Rect(GraphRect.Left-vWidth-dx, GraphRect.Bottom-vY-
(vHeight div 2),
      GraphRect.Left-dx, GraphRect.Bottom-vY-(vHeight div
2)+vHeight);
    TextDraw(Image1.Canvas, vTeks,
DT_CENTER+DT_VCENTER+DT_SINGLELINE, vRect);
    vY := Round(pAlt^.Nilai1*(GraphRect.Bottom-GraphRect.Top));
    Image1.Canvas.LineTo(GraphRect.Right, GraphRect.Bottom-vY);
    vTeks := FormatFloat(FloatFormat, pAlt^.Nilai1);
    vRect := Rect(GraphRect.Right+dx, GraphRect.Bottom-vY-(vHeight
div 2),
      GraphRect.Right+dx+vWidth, GraphRect.Bottom-vY-(vHeight div
2)+vHeight);
    TextDraw(Image1.Canvas, vTeks,
DT_CENTER+DT_VCENTER+DT_SINGLELINE, vRect);
  end;
  Image1.Canvas.Brush.Color := clWhite;
  Image1.Canvas.Brush.Style := bsSolid;
end;
```

```
procedure TFormSensGraph.ComboBox1Change(Sender: TObject);
begin
  RefreshLegend;
  RefreshGraph;
end;

procedure TFormSensGraph.ImageClick(Sender: TObject);
begin
  if Sender is TImage then begin
    SelectNumber := TImage(Sender).Tag;
    ShowCurrentSelect;
    DrawHint;
  end;
end;

procedure TFormSensGraph.DrawHint;
var vRect: TRect;
  vTeks: string;
  pAlt: pTItemAlt;
  vNilai: Extended;
begin
  if SelectNumber = -1 then begin
    vTeks := 'X: '+FormatFloat('0.00', VertLinePos);
    ImageHint.Width := FWidth div 2;
  end else begin
    vTeks := 'X: '+FormatFloat('0.00', VertLinePos);
    if (SelectNumber<FListAlt.Count) then begin
      {Kriteria.HitungSensValue(VertLinePos);
      RootKriteria.HitungPreferensi(Tree.DoHitungKriteria);
      Tree.HitungAlternative;}
      pAlt := FListAlt[SelectNumber];
      vNilai := 0.00;
      vTeks := vTeks+' Y: '+FormatFloat('0.00', vNilai);
      ImageHint.Width := FWidth;
    end else begin
      ImageHint.Width := FWidth div 2;
    end;
  end;
  ImageHint.Canvas.Brush.Color := clYellow;
  vRect := Rect(0, 0, ImageHint.Width, ImageHint.Height);
  ImageHint.Canvas.Brush.Style := bsSolid;
  ImageHint.Canvas.Rectangle(0, 0, ImageHint.Width,
ImageHint.Height);
  ImageHint.Canvas.Brush.Style := bsClear;
  TextDraw(ImageHint.Canvas, vTeks,
DT_VCENTER+DT_CENTER+DT_SINGLELINE, vRect);
  ImageHint.Top := Image1.Top+Image1.Height-GraphMargin.Bottom+10;
  ImageHint.Refresh;
  ImageHint.Left := VertLine.Left-(ImageHint.Width div 2)
end;
```

```
procedure TFormSensGraph.ShowCurrentSelect;
var pAlt: pTItemAlt;
begin
  if (SelectNumber < 0) or (SelectNumber>=FListAlt.Count) then begin
    Shape2.Visible := False;
  end else begin
    pAlt := FListAlt[SelectNumber];
    Shape2.Left := pAlt^.AltImage.Left-2;
    Shape2.Top := pAlt^.AltImage.Top-2;
    Shape2.Width := pAlt^.AltImage.Width+4;
    Shape2.Height := pAlt^.AltImage.Height+4;
    Shape2.Visible := True;
  end;
end;

procedure TFormSensGraph.SetVertLine;
var vWidth: Integer;
begin
  vWidth := Image1.Width-GraphMargin.Left-GraphMargin.Right;
  VertLine.Left :=
Image1.Left+GraphMargin.Left+Round(VertLinePos*vWidth);
  VertLine.Top := Image1.Top + GraphMargin.Top;
  VertLine.Height := Image1.Height- GraphMargin.Top-
GraphMargin.Bottom;
end;


procedure TFormSensGraph.ScrollBox1Click(Sender: TObject);
begin
  SelectNumber := -1;
  ShowCurrentSelect;
  DrawHint;
end;

procedure TFormSensGraph.FormResize(Sender: TObject);
begin
  if (Width < 400) or (Height<300) then begin
    if Width <400 then begin
      Width := 400;
    end;
    if Height <300 then begin
      Height := 300;
    end;
  end else begin
    RefreshLegend;
    RefreshGraph;
    ShowCurrentSelect;
    SetVertLine;
    DrawHint;
  end;
end;

procedure TFormSensGraph.Image1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var vWidth: Integer;
begin
```

```
  if (X>GraphMargin.Left) and (X<(Image1.Width-GraphMargin.Right))
and
    (Y>GraphMargin.Top) and (Y<Image1.Height-GraphMargin.Bottom)
then begin
    vWidth := Image1.Width-GraphMargin.Left-GraphMargin.Right;
    DownState := True;
    VertLinePos := (X-GraphMargin.Left)/vWidth;
    SetVertLine;
    DrawHint;
  end;
end;

procedure TFormSensGraph.Image1MouseMove(Sender: TObject;
  Shift: TShiftState; X, Y: Integer);
var vWidth: Integer;
begin
  if DownState then begin
    vWidth := Image1.Width-GraphMargin.Left-GraphMargin.Right;
    VertLinePos := (X-GraphMargin.Left)/vWidth;
    if VertLinePos<0 then begin
      VertLinePos := 0;
    end;
    if VertLinePos>1 then begin
      VertLinePos := 1;
    end;
    SetVertLine;
    DrawHint;
  end;
end;

procedure TFormSensGraph.Image1MouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  DownState := False;
end;

procedure TFormSensGraph.ComboBox2Change(Sender: TObject);
var vName: string;
begin
  if FCalcFromProgram then exit;
  if ComboBox1.ItemIndex>=0 then begin
    vName := ComboBox2.Items[ComboBox2.ItemIndex];
    SetParentByName(vName);
    CalculatingData;
  end;
end;

procedure TFormSensGraph.SetParentByName(vName: string);
var idx: Integer;
  vBlock: TKBlockObject;
begin
  idx := FDataStru.FindCriteria(vName);
  vBlock := nil;
  try
    vBlock := FDataStru.CritList[idx];
    if vBlock.FieldCount=0 then begin
```

```
      vBlock := nil;
    end;
  except
    vBlock := nil;
  end;
  if vBlock <> nil then begin
    FSelected := vBlock;
    RefreshGraphData;
    RefreshGraph;
  end;
end;

procedure TFormSensGraph.RefreshGraphData;
var vParent: TKBlockObject;
  vJumlah, vJumlah1: Extended;
  vParentValue: Extended;
  i: Integer;
  vField: TKFieldObject;
  pData: pTItemAlt;
  procedure Hitung;
  var i: Integer;
  begin
    vJumlah := 0;
    vJumlah1 := 0;
    vParentValue := vParent.TempReal3;
    for i:=0 to vParent.FieldCount-1 do begin
      vField := vParent[i];
      vField :=
TKBlockField(vField).BlockData.FieldByName('Criteria');
      vJumlah := vJumlah+TKBlockField(vField).BlockData.TempReal4;
      vJumlah1 := vJumlah1+TKBlockField(vField).BlockData.TempReal4;
    end;
    for i:=0 to vParent.FieldCount-1 do begin
      vField := vParent[i];
      vField :=
TKBlockField(vField).BlockData.FieldByName('Criteria');
      if vJumlah1 = 0 then begin
        raise Exception.Create('Pembagian dengan Nol');
      end else begin
        TKBlockField(vField).BlockData.TempReal4 :=
          TKBlockField(vField).BlockData.TempReal4*vJumlah/vJumlah1;
        TKBlockField(vField).BlockData.TempReal3 :=
          TKBlockField(vField).BlockData.TempReal4*vParentValue;
      end;
    end;
    FDataStru.TraceTree(nil, postProcess, nil, True);
    FDataStru.HitungAllAlternative;
  end;
begin
  if FSelected <> nil then begin
    vParent := FSelected.OwnerField.Owner.OwnerField.Owner;
    FSelected.TempReal4 := 0;
    Hitung;
    for i:=0 to FDataStru.AltList.FieldCount-1 do begin
      vField := FDataStru.AltList[i];
      pData := FListAlt[i];
```

```
      pData^.Nilai0 := TKBLockField(vFIeld).BlockData.TempReal1;
    end;
    FSelected.TempReal4 := 1;
    Hitung;
    for i:=0 to FDataStru.AltList.FieldCount-1 do begin
      vField := FDataStru.AltList[i];
      pData := FListAlt[i];
      pData^.Nilai1 := TKBLockField(vFIeld).BlockData.TempReal1;
    end;
  end;
end;

procedure TFormSensGraph.PostProcess(vCrit: TKBlockObject;
  vLevel: Integer);
var vx, vy: Integer;
  vFIeld: TKFieldObject;
  i: Integer;
  vParentValue: Extended;
begin
  if vCrit.OwnerField.Owner.OwnerField = nil then begin
    vParentValue := 1;
  end else begin
    vParentValue :=
vCrit.OwnerField.Owner.OwnerField.Owner.TempReal3;
  end;
  vCrit.TempReal3 := vCrit.TempReal4*vParentValue;
end;

procedure TFormSensGraph.FormShow(Sender: TObject);
begin
    ComboBox2.ItemIndex := 0;
    ComboBox2Change(ComboBox2);
end;

end.
```

```
unit Tokutils;

interface

uses WinProcs, WinTypes, SysUtils, Graphics;

procedure TextDraw(Canvas: TCanvas; S: string; Style: Word; xRect:
TRect);
procedure RGBToHLS(vRGB: Longint; var vH, vL, vS: Byte);
function HLSToRGB(vH, vL, vS: Byte): Longint;
function LPad(vData: string; Width: integer): string;
function RPad(vData: string; Width: integer): string;

implementation

procedure TextDraw(Canvas: TCanvas; S: string; Style: Word; xRect:
TRect);
var xchar: Array[0..255] of Char;
begin
  StrPCopy(xchar, S);
  DrawText(Canvas.Handle, xchar, -1, xRect, Style);
end;


procedure RGBToHLS(vRGB: Longint; var vH, vL, vS: Byte);
type
  TXRGB = record
    case Integer of
      0: (Color: Longint);
      1: (cR, cG, cB, cN: byte);
  end;
var vMax, vMin: real;
  delta: real;
  r,g,b: real;
  h, l, s: real;
  vxRGB: TXRGB;
begin
  vxRGB.Color := vRGB;
  r := vxRGB.cR/255;
  g := vxRGB.cG/255;
  b := vxRGB.cB/255;
  if r>g then begin
    vMax := r;
    vMin := g;
  end else begin
    vMax := g;
    vMin := r;
  end;
  if vMax < b then begin
    vMax := b;
  end;
  if vMin > b then begin
    vMin := b;
  end;
  l := (vMax+vMin)/2;
  if vMax = vMin then begin
```

```
    s := 0;
    h := 255;
  end else begin
    if l<=0.5 then begin
      s := (vMax-vMin)/(vMax+vMin);
    end else begin
      s := (vMax-vMin)/(2-vMax-vMin);
    end;
    delta := vMax-vMin;
    if r = vMax then begin
      h := (g-b)/delta;
    end else if g = vMax then begin
      h := 2+(b-r)/delta;
    end else begin
      h := 4+(r-g)/delta;
    end;
    h := h*40;
    if h<0 then begin
      h := h+240;
    end;
  end;
  (*  sampai disini nilai l dan s pada [0,1],
      nilai h [0,240) *)
  vH := Round(h);
  vL := Round(l*255);
  vS := Round(s*255);
end;

function HLSToRGB(vH, vL, vS: Byte): Longint;
type
  TXRGB = record
    case Integer of
      0: (Color: Longint);
      1: (cR, cG, cB, cN: byte);
  end;
var vMax, vMin: real;
  delta: real;
  r,g,b: real;
  h, l, s: real;
  vxRGB: TXRGB;
  m1, m2: real;
  function Value(n1, n2, hue: real): real;
  begin
    if hue>240 then begin
      hue := hue - 240;
    end else if hue<0 then begin
      hue := hue+240;
    end;
    if hue<40 then begin
      result := n1 + (n2-n1)*hue/40;
    end else if hue<120 then begin
      result := n2;
    end else if hue<160 then begin
      result := n1 + (n2-n1)*(160-hue)/40;
    end else begin
      result := n1;
```

```
    end;
  end;
begin
  h := vH;
  if h>240 then h := 240;
  l := vL/255;
  s := vs/255;
  if l<= 0.5 then begin
    m2 := l*(1+s);
  end else begin
    m2 := l + s - l*s;
  end;
  m1 := 2*l - m2;
  if s = 0 then begin
    r := l;
    g := l;
    b := l;
  end else begin
    r := Value(m1, m2, h+80);
    g := value(m1, m2, h);
    b := value(m1, m2, h-80);
  end;
  vxRGB.cR := Round(r*255);
  vxRGB.cG := Round(g*255);
  vxRGB.cB := Round(b*255);
  vxRGB.cN := 0;
  result := vxRGB.Color;
end;

function LPad(vData: string; Width: integer): string;
var vx: string;
  i: Integer;
begin
  result := vData;
  for i:=1 to (Width-Length(vData)) do begin
    result := ' '+result;
  end;
end;

function RPad(vData: string; Width: integer): string;
var vx: string;
  i: Integer;
begin
  result := vData;
  for i:=1 to (Width-Length(vData)) do begin
    result := result+' ';
  end;
end;

end.
```

```
unit Fprogln;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, StdCtrls, ExtCtrls, Grids, {DTree, }Tabs, Gomory,
  KFieldObjects, UDataStru, GLobalUnit;

type
  TFormProgramaLinier = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    LabelMax: TLabel;
    Panel3: TPanel;
    Button1: TButton;
    Button3: TButton;
    RadioGroup1: TRadioGroup;
    Panel4: TPanel;
    TabSet1: TTabSet;
    Notebook1: TNotebook;
    StringGrid1: TStringGrid;
    Memo1: TMemo;
    Bevel1: TBevel;
    Memo2: TMemo;
    Panel2: TPanel;
    Button4: TButton;
    Button5: TButton;
    ComboBox1: TComboBox;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure TabSet1Change(Sender: TObject; NewTab: Integer;
      var AllowChange: Boolean);
    procedure Button5Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure StringGrid1MouseUp(Sender: TObject; Button:
TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure ComboBox1Change(Sender: TObject);
    procedure ComboBox1Exit(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure StringGrid1DrawCell(Sender: TObject; ACol, ARow:
Integer;
      xRect: TRect; State: TGridDrawState);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
    JumlahAlternatif: Longint;
    FDataStru: TRDataStru;
    FListAltName: TStringList;
  public
    { Public declarations }
    //MyTree: TTreeContainer;
```

```
    //KritList: TList;
    //AltList: TList;
    ListKendala: TStringList;
    //procedure CallBackKritList(Element: TTreeElement; var Stop:
Boolean);
    //procedure CallBackAltList(Element: TTreeElement; var Stop:
Boolean);
    procedure SetDataStru(vDataStru: TRDataStru);
    procedure ListKriteria;
    procedure RevSimplexProcess;
    procedure GomorySimplexProcess;
    procedure Genetik;
    procedure SaveData;
  end;

var
  FormProgramaLinier: TFormProgramaLinier;

implementation

uses RevSplx;
{$R *.DFM}

procedure TFormProgramaLinier.Button1Click(Sender: TObject);
begin
  Memo1.Clear;
  Memo2.Clear;
  case RadioGroup1.ItemIndex of
    0: RevSimplexProcess;
    1: GomorySimplexProcess;
    2: Genetik;
  end;
end;

procedure TFormProgramaLinier.RevSimplexProcess;
var MySimplex: TRevSimplex;
  vgc, vec, vlc: Integer;
  vHasil: Extended;
  vCode: Integer;
  ctr: Integer;
  i,j: Integer;
begin
  try
  MySimplex := TRevSimplex.Create;
  MySimplex.HasilDetilMemo := Memo2;
  MySimplex.HasilMemo := Memo1;
  MySimplex.ClearOutput;
{  MySimplex.InputData;}
  MySimplex.nvar :=  FListAltName.Count;
  MySimplex.m := StringGrid1.RowCount-1;
  MySimplex.ncols := MySimplex.nvar + MySimplex.m;
  vgc := 0;
  vec := 0;
  vlc := 0;
  ctr := 0;
  for i:=1 to StringGrid1.RowCount-1 do begin
```

```
    if StringGrid1.Cells[FListAltName.Count+1, i] = '>=' then begin
      vgc := vgc+1;
      ctr := ctr+1;
      for j:=1 to FListAltName.Count do begin
        Val(StringGrid1.Cells[j, i], vHasil, vCode);
        if vCode = 0 then begin
          MySimplex.a[ctr, j] := vHasil;
        end else begin
          MySimplex.a[ctr, j] := 0;
        end;
      end;
      Val(StringGrid1.Cells[FListAltName.Count+2, i], vHasil,
vCode);
      if vCode = 0 then begin
        MySimplex.b[ctr] := vHasil;
      end else begin
        MySimplex.b[ctr] := vHasil;
      end;
    end;
  end;
  for i:=1 to StringGrid1.RowCount-1 do begin
    if StringGrid1.Cells[FListAltName.Count+1, i] = '=' then begin
      vec := vec+1;
      ctr := ctr+1;
      for j:=1 to FListAltName.Count do begin
        Val(StringGrid1.Cells[j, i], vHasil, vCode);
        if vCode = 0 then begin
          MySimplex.a[ctr, j] := vHasil;
        end else begin
          MySimplex.a[ctr, j] := 0;
        end;
      end;
      Val(StringGrid1.Cells[FListAltName.Count+2, i], vHasil,
vCode);
      if vCode = 0 then begin
        MySimplex.b[ctr] := vHasil;
      end else begin
        MySimplex.b[ctr] := vHasil;
      end;
    end;
  end;
  for i:=1 to StringGrid1.RowCount-1 do begin
    if StringGrid1.Cells[FListAltName.Count+1, i] = '<=' then begin
      vlc := vlc+1;
      ctr := ctr+1;
      for j:=1 to FListAltName.Count do begin
        Val(StringGrid1.Cells[j, i], vHasil, vCode);
        if vCode = 0 then begin
          MySimplex.a[ctr, j] := vHasil;
        end else begin
          MySimplex.a[ctr, j] := 0;
        end;
      end;
      Val(StringGrid1.Cells[FListAltName.Count+2, i], vHasil,
vCode);
      if vCode = 0 then begin
```

```
        MySimplex.b[ctr] := vHasil;
      end else begin
        MySimplex.b[ctr] := vHasil;
      end;
    end;
  end;
  MySimplex.gc := vgc;
  MySimplex.ec := vec;
  MySimplex.lc := vlc;
  for i:=0 to FListAltName.Count-1 do begin
    //MySimplex.c[i+1] := -TTreeElement(AltList[i]).Preferensi;
    MySimplex.c[i+1] := -
TKBLockField(FDataStru.AltList[i]).BlockData.TempReal1;
    //--
  end;
  MySimplex.RevisedSimplex;
  Memo1.Clear;
  if MySimplex.Takterbatas then begin
    Memo1.Lines.Add(' Hasil Tak Terbatas ');
  end else begin
    Memo1.Lines.Add(' Hasil Perhitungan ');
    Memo1.Lines.Add('');
    for i:=1 to MySimplex.m do begin
      //Memo1.Lines.Add(TTreeElement(AltList[i-1]).DataName+':
'+FormatFloat('###0.###', MySimplex.Hasil[i]));

Memo1.Lines.Add(TKBLockField(FDataStru.AltList[i]).FieldName+': '+
        FormatFloat('###0.###', MySimplex.Hasil[i-1]));
      //--
    end;
  end;
  MySimplex.Destroy;
  except
    Memo1.Clear;
    Memo1.Lines.Add('Process Error');
    Memo2.Clear;
    Memo2.Lines.Add('Process Error');
  end;
end;

procedure TFormProgramaLinier.GomorySimplexProcess;
var MySimplex: TGomorySimplex;
  vgc, vec, vlc: Integer;
  vHasil: Extended;
  vCode: Integer;
  ctr: Integer;
  i,j: Integer;
begin
  try
  MySimplex := TGomorySimplex.Create;
  MySimplex.HasilDetilMemo := Memo2;
  MySimplex.HasilMemo := Memo1;
  MySimplex.ClearOutput;
  {MySimplex.InputData;}
  MySimplex.nvar := FListAltName.Count;
  MySimplex.m := StringGrid1.RowCount-1;
```

```
  {MySimplex.ncols := MySimplex.nvar + MySimplex.m;}
  vgc := 0;
  vec := 0;
  vlc := 0;
  ctr := 0;
  for i:=1 to StringGrid1.RowCount-1 do begin
    if StringGrid1.Cells[FListAltName.Count+1, i] = '>=' then begin
      vgc := vgc+1;
      ctr := ctr+1;
      for j:=1 to FListAltName.Count do begin
        Val(StringGrid1.Cells[j, i], vHasil, vCode);
        if vCode = 0 then begin
          MySimplex.a[ctr, j] := vHasil;
        end else begin
          MySimplex.a[ctr, j] := 0;
        end;
      end;
      Val(StringGrid1.Cells[FListAltName.Count+2, i], vHasil,
vCode);
      if vCode = 0 then begin
        MySimplex.b[ctr] := vHasil;
      end else begin
        MySimplex.b[ctr] := vHasil;
      end;
    end;
  end;
  for i:=1 to StringGrid1.RowCount-1 do begin
    if StringGrid1.Cells[FListAltName.Count+1, i] = '=' then begin
      vec := vec+1;
      ctr := ctr+1;
      for j:=1 to FListAltName.Count do begin
        Val(StringGrid1.Cells[j, i], vHasil, vCode);
        if vCode = 0 then begin
          MySimplex.a[ctr, j] := vHasil;
        end else begin
          MySimplex.a[ctr, j] := 0;
        end;
      end;
      Val(StringGrid1.Cells[FListAltName.Count+2, i], vHasil,
vCode);
      if vCode = 0 then begin
        MySimplex.b[ctr] := vHasil;
      end else begin
        MySimplex.b[ctr] := vHasil;
      end;
    end;
  end;
  for i:=1 to StringGrid1.RowCount-1 do begin
    if StringGrid1.Cells[FListAltName.Count+1, i] = '<=' then begin
      vlc := vlc+1;
      ctr := ctr+1;
      for j:=1 to FListAltName.Count do begin
        Val(StringGrid1.Cells[j, i], vHasil, vCode);
        if vCode = 0 then begin
          MySimplex.a[ctr, j] := vHasil;
        end else begin
```

```
          MySimplex.a[ctr, j] := 0;
        end;
      end;
      Val(StringGrid1.Cells[FListAltName.Count+2, i], vHasil,
vCode);
      if vCode = 0 then begin
        MySimplex.b[ctr] := vHasil;
      end else begin
        MySimplex.b[ctr] := vHasil;
      end;
    end;
  end;
  MySimplex.gc := vgc;
  MySimplex.ec := vec;
  MySimplex.lc := vlc;
  for i:=0 to FListAltName.Count-1 do begin
    //MySimplex.c[i+1] := -TTreeElement(AltList[i]).Preferensi;
    MySimplex.c[i+1] := -
TKBLockField(FDataStru.AltList[i]).BlockData.TempReal1;
    //--
  end;
  MySimplex.SimplexProcess;
  Memo1.Clear;
  if MySimplex.Takterbatas then begin
    Memo1.Lines.Add(' Hasil Tak Terbatas ');
  end else begin
    Memo1.Lines.Add(' Hasil Perhitungan ');
    Memo1.Lines.Add('');
    for i:=1 to FListAltName.Count do begin
      //Memo1.Lines.Add(TTreeElement(AltList[i-1]).DataName+':
'+FormatFloat('###0.###', MySimplex.Hasil[i]));
      Memo1.Lines.Add(TKBLockField(FDataStru.AltList[i-
1]).FieldName+': '+
        FormatFloat('###0.###', MySimplex.Hasil[i]));
      //--
    end;
  end;
  MySimplex.Destroy;
  except
    Memo1.Clear;
    Memo1.Lines.Add('Process Error');
    Memo2.Clear;
    Memo2.Lines.Add('Process Error');
  end;
end;

procedure TFormProgramaLinier.Genetik;
  function DuaPangkat(n: Integer):Longint;
    var i: integer;
    vhasil: Longint;
  begin
    vHasil := 1;
    for i:=1 to n do begin
      vHasil := vHasil*2;
    end;
    result := vHasil;
```

```
  end;
  function GetPosisi(Data:Longint; Posisi: Integer): Boolean;
  begin
    result := (Data shr (Posisi-1)) and 1 = 1;
  end;
var i, j, k: Integer;
  Id: Integer;
  vMax: Extended;
  vValue, vKendala: Extended;
  vId: Longint;
  vMaxId: Longint;
  vKode: Integer;
  vMaxCol: Integer;
  vMaxRow: Integer;
  vKendalaPos: Integer;
  vParam: Extended;
  Terus: Boolean;
  vMaksimum: Extended;
begin
  Memo1.Clear;
  Memo2.Clear;
  vMaxId := DuaPangkat(FListAltName.Count);
  vMaxCol := StringGrid1.ColCount-3;
  vMaxRow := StringGrid1.RowCount-1;
  if (vMaxRow<=0) or (vMaxCol<=0) then begin
    exit;
  end;
  vKendalaPos := StringGrid1.ColCount-1;
  vMax := -1E20;
  vId := -1;
  for id:=0 to vMaxId-1 do begin
    Terus := True;
    j := 1;
    while (j<=StringGrid1.RowCount-1) and Terus do begin
      Val(StringGrid1.Cells[vKendalaPos, j], vKendala, vKode);
      if vKode<>0 then begin
        vKendala := 0;
      end;
      vValue := 0;
      k:=1;
      for k:=1 to StringGrid1.ColCount-3 do begin
        if GetPosisi(Id, k) then begin
          Val(StringGrid1.Cells[k, j], vParam, vKode);
          if vKode=0 then begin
            vValue := vValue+vParam;
          end;
        end;
      end;
      if StringGrid1.Cells[vKendalaPos-1, j] = '<=' then begin
        if not (vValue<=vKendala) then begin
          Terus := False;
        end;
      end else if StringGrid1.Cells[vKendalaPos-1, j] = '=' then
begin
        if not (vValue=vKendala) then begin
          Terus := False;
```

```
        end;
      end else if StringGrid1.Cells[vKendalaPos-1, j] = '>=' then
begin
        if not (vValue>=vKendala) then begin
          Terus := False;
        end;
      end else begin
        Terus := False;
      end;
      inc(j);
    end;
    if Terus then begin
      vMaksimum := 0;
      for k:=0 to FListAltName.Count-1 do begin
        if GetPosisi(Id, k+1) then begin
          //vMaksimum :=
vMaksimum+TTreeElement(AltList[k]).Preferensi;
        end;
      end;
      if vMax<vMaksimum then begin
        vMax := vMaksimum;
        vId := Id;
      end;
    end;
  end;
  Memo1.Lines.Add('Penghitungan Programa Linier Bilangan 0-1');
  Memo1.Lines.Add('Dengan Metode Algoritma Genetik');
  if vId=-1 then begin
    Memo1.Lines.Add('Tidak Ada Yang Memenuhi');
  end else begin
    vMaksimum := 0;
    for k:=0 to FListAltName.Count-1 do begin
      if GetPosisi(vId, k+1) then begin
        //vMaksimum :=
vMaksimum+TTreeElement(AltList[k]).Preferensi;
      end;
    end;
    Memo1.Lines.Add('Nilai Maksimum Adalah :
'+FormatFloat('#####0.00', vMaksimum));
    for k:=0 to FListAltName.Count-1 do begin
      if GetPosisi(vId, k+1) then begin
        //Memo1.Lines.Add(TTreeElement(AltList[k]).DataName+' = 1');
      end else begin
        //Memo1.Lines.Add(TTreeElement(AltList[k]).DataName+' = 0');
      end;
    end;
  end;
end;

{procedure TFormProgramaLinier.CallBackKritList(Element:
TTreeElement; var Stop: Boolean);
begin
  if (Element.FirstChild=nil) and (Element.Parent<>nil) then begin
    KritList.Add(Element);
  end;
end;
```

```
}
{procedure TFormProgramaLinier.CallBackAltList(Element:
TTreeElement; var Stop: Boolean);
begin
  if (Element.FirstChild=nil) and (Element.Parent<>nil) then begin
    AltList.Add(Element);
  end;
end;
}
procedure TFormProgramaLinier.ListKriteria;
var Stop: Boolean;
begin
  //FListAltName.Clear;
  //AltList.Clear;
{  if MyTree<>nil then begin
    Stop := False;
    MyTree.RootElement.Trace(CallBackKritList, Stop);
    Stop := False;
    MyTree.RootAlternative.Trace(CallBackAltList, Stop);
  end;}
end;

procedure TFormProgramaLinier.FormCreate(Sender: TObject);
begin
  FListAltName := TStringLIst.Create;
  //FListAltName := TList.Create;
  //KritList := TList.Create;
  StringGrid1.RowCount := 2;
  ListKendala := TStringList.Create;
  JumlahAlternatif := 0;
end;

procedure TFormProgramaLinier.FormDestroy(Sender: TObject);
begin
  FListAltName.Free;
  //AltList.Destroy;
  //KritList.Destroy;
  ListKendala.Destroy;
end;

procedure TFormProgramaLinier.FormShow(Sender: TObject);
var vStr: string;
  i: Integer;
  MyFormat: String;
  //pData: TTreeElement;
  xStr: string;
begin
{  TabSet1.TabIndex := 0;
  MyFormat := '##0.###';
  vStr := '';
  for i:=0 to AltList.Count-1 do begin
    pData := AltList[i];
    xStr := FormatFloat(MyFormat,
pData.Preferensi)+'*'+pData.DataName+'';
    if vStr = '' then begin
      vStr := xStr;
```

```
    end else begin
      vStr := vStr+'+'+xStr;
    end;
  end;
  LabelMax.Caption := vStr;
  StringGrid1.ColCount := AltList.Count+3;
  if JumlahAlternatif <> 0 then begin
    if JumlahAlternatif<AltList.Count then begin

StringGrid1.Cols[AltList.Count+1].Assign(StringGrid1.Cols[JumlahAlte
rnatif+1]);
      StringGrid1.Cols[JumlahAlternatif+1].Clear;
    end;
  end;
  JumlahAlternatif := AltList.Count;
  for i:=0 to AltList.Count-1 do begin
    pData := AltList[i];
    StringGrid1.Cells[i+1, 0] := pData.DataName;
  end;
  StringGrid1.Cells[0, 0] := 'Kendala';
  StringGrid1.Cells[AltList.Count+1, 0] := 'Tanda';
  StringGrid1.Cells[AltList.Count+2, 0] := 'Pembatas';}
end;

procedure TFormProgramaLinier.TabSet1Change(Sender: TObject;
  NewTab: Integer; var AllowChange: Boolean);
begin
  NoteBook1.PageIndex := NewTab;
end;

procedure TFormProgramaLinier.Button5Click(Sender: TObject);
var i: Integer;
  vStrList: TStringList;
  Baris: Integer;
begin
  if ListKendala.Count = 1 then begin
    ListKendala.Clear;
    for i:=0 to StringGrid1.ColCount-1 do begin
      StringGrid1.Cells[i, 1] := '';
    end;
  end else if ListKendala.Count > 1 then begin
    Baris := StringGrid1.Row;
    for i:=Baris+1 to StringGrid1.RowCount-1 do begin
      StringGrid1.Rows[i-1] := StringGrid1.Rows[i];
    end;
    StringGrid1.RowCount := StringGrid1.RowCount-1;
    ListKendala.Delete(Baris-1);
  end;
end;

procedure TFormProgramaLinier.Button4Click(Sender: TObject);
var vTeks: string;
begin
  vTeks := '';
  if InputQuery('Insert Kendala ', 'Id Kendala : ', vTeks) then
begin
```

```
    if ListKendala.Count = 0 then begin
      StringGrid1.Cells[0, 1] := vTeks;
      ListKendala.Add(vteks);
    end else begin
      StringGrid1.RowCount := StringGrid1.RowCount+1;
      StringGrid1.Row := StringGrid1.RowCount-1;
      StringGrid1.Cells[0, StringGrid1.RowCount-1] := vTeks;
      ListKendala.Add(vTeks);
    end;
    StringGrid1.Cells[StringGrid1.ColCount-2, StringGrid1.Row] :=
'<=';
  end;
end;

procedure TFormProgramaLinier.StringGrid1MouseUp(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
var vCol, vRow: Longint;
  vRect: TRect;
begin
  if ListKendala.Count = 0 then exit;
  StringGrid1.MouseToCell(X, Y, vCol, vRow);
  if (vCol = StringGrid1.ColCount-2) and (vRow>0) then begin
    ComboBox1.Text := StringGrid1.Cells[vCol, vRow];
    vRect := StringGrid1.CellRect(vCol, vRow);
    vRect.TopLeft := ComboBox1.Parent.ScreenToClient(
      StringGrid1.ClientToScreen(vRect.TopLeft));
    ComboBox1.Left := vRect.Left;
    ComboBox1.Top := vRect.Top;
    ComboBox1.Visible := True;
    ComboBox1.SetFocus;
  end;

end;

procedure TFormProgramaLinier.ComboBox1Change(Sender: TObject);
begin
  StringGrid1.Cells[StringGrid1.Col, StringGrid1.Row] :=
ComboBox1.Text;
end;

procedure TFormProgramaLinier.ComboBox1Exit(Sender: TObject);
begin
  ComboBox1.Visible := False;
end;

procedure TFormProgramaLinier.Button3Click(Sender: TObject);
begin
  Close;
end;

procedure TFormProgramaLinier.SetDataStru(vDataStru: TRDataStru);
var i, j: Integer;
  vFIeld: TKFieldObject;
  vStrList: TStringList;
  vBlock: TKBlockObject;
  vTanda: string;
```

```
  vPembatas: string;
  vBatas: Integer;
  vStr: string;
  vListName: TStringList;
  vPos: Integer;
  MyFormat: string;
  xStr: string;
begin
  FListAltName.Clear;
  FDataStru := vDataStru;
  for i:=0 to FDataStru.AltList.FieldCount-1 do begin
    vField := FDataStru.AltList[i];
    FListAltName.Add(vFIeld.FieldName);
  end;
  StringGrid1.ColCount := FListAltName.Count+3;
  for i:=0 to FListAltName.Count-1 do begin
    StringGrid1.Cells[i+1, 0] := FListAltName[i];
  end;

  MyFormat := '##0.###';
  vStr := '';
  for i:=0 to FDataStru.AltList.FieldCount-1 do begin
    vField := FDataStru.AltList[i];
    xStr := FormatFloat(MyFormat,
TKBlockField(vField).BlockData.TempReal1)+
      '*'+vField.FieldName+'';
    if vStr = '' then begin
      vStr := xStr;
    end else begin
      vStr := vStr+'+'+xStr;
    end;
  end;
  LabelMax.Caption := vStr;

  StringGrid1.Cells[0,0] := 'Kendala';
  StringGrid1.Cells[FListAltName.Count+1,0] := 'Tanda';
  StringGrid1.Cells[FlistAltName.Count+2,0] := 'Pembatas';
  vStrList := TStringList.Create;
  vListName := TStringList.Create;
  try
    vField := FDataStru.StruData.FieldByName('LinearProgData');
    vBlock := TKBlockField(vField).BlockData;
    if (vBlock.FieldCount=0) or (vBlock.FieldCount=1) then begin
      StringGrid1.RowCount := 2;
      StringGrid1.Rows[1].Text := '';
    end else begin
      StringGrid1.RowCount := vBlock.FieldCount;
    end;
    vBatas := FListAltName.Count+1;
    if vBlock.FieldCount>0 then begin
      vField := vBlock[0];
      if vField is TKStringField then begin
        vListName.Text := TKStringField(vField).StringData;
      end;
    end;
    for i:=1 to StringGrid1.RowCount-1 do begin
```

```
      StringGrid1.Rows[i].Text := '';
    end;
    for i:=1 to vBlock.FieldCount-1 do begin
      vField := vBlock[i];
      if vField is TKStringField then begin
        vStrList.Text := TKStringField(vField).StringData;
        for j:=0 to FListAltName.Count-1 do begin
          vPos := vListName.IndexOf(FListAltName[j]);
          if (vPos>0) and (vPos<StringGrid1.ColCount-2) then begin
            StringGrid1.Cells[vPos, i] := vStrList[vPos];
          end;
        end;
        StringGrid1.Cells[0, i] := vField.FieldName;
        vPos := vListName.IndexOf('Tanda');
        if (vPos>0) then begin
          StringGrid1.Cells[StringGrid1.ColCount-2, i] :=
vStrList[vPos];
        end;
        vPos := vListName.IndexOf('Pembatas');
        if (vPos>0) then begin
          StringGrid1.Cells[StringGrid1.ColCount-1, i] :=
vStrList[vPos];
        end;
      end;
    end;
  finally
    vStrList.Free;
    vListName.Free;
  end;
end;

procedure TFormProgramaLinier.StringGrid1DrawCell(Sender: TObject;
ACol,
  ARow: Integer; xRect: TRect; State: TGridDrawState);
var vTeks: string;
begin
  if ARow = 0 then begin
    if (ACol>0) and (ACol<=StringGrid1.ColCount-3) then begin
      vTeks := FListAltName[ACol-1];
    end else begin
      if ACol = 0 then begin
        vTeks := 'Kendala';
      end else if ACol=StringGrid1.ColCount-2 then begin
        vTeks := 'Tanda';
      end else begin
        vTeks := 'Pembatas';
      end;
    end;
  end;
  xRect.Left := xRect.Left+2;
  TextDraw(StringGrid1.Canvas, vTeks, DT_SINGLELINE+DT_VCENTER,
xRect);
end;

procedure TFormProgramaLinier.SaveData;
var vBlock: TKBLockObject;
```

```
  vField: TKFieldObject;
  i: Integer;
  vTeks: string;
begin
  vField := FDataStru.StruData.FieldByName('LinearProgData');
  vBlock := TKBlockField(vField).BlockData;
  for i:=vBlock.FieldCOunt-1 downto 0 do begin
    vField := vBlock[i];
    vField.Free;
  end;
  vField := vBlock.AddField(CFIELD_STRING, vTeks);
  TKStringField(vFIeld).StringData := StringGrid1.Rows[0].Text;
  for i:=1 to StringGrid1.RowCount-1 do begin
    vTeks := StringGrid1.Cells[0, i];
    if vTeks <> '' then begin
      vField := vBlock.AddField(CFIELD_STRING, vTeks);
      TKStringField(vFIeld).StringData := StringGrid1.Rows[i].Text;
    end;
  end;
end;

procedure TFormProgramaLinier.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  SaveData;
end;

end.
```

```
unit Splash;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, ExtCtrls, StdCtrls, Gauges;

type
  TFormSplash = class(TForm)
    Image1: TImage;
    Timer1: TTimer;
    Gauge1: TGauge;
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Image1Click(Sender: TObject);
    procedure FormKeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormSplash: TFormSplash;

implementation
{$R *.DFM}

procedure TFormSplash.FormCreate(Sender: TObject);
begin
  ClientWidth := Image1.Width;
  ClientHeight := Image1.Height;
end;

procedure TFormSplash.Timer1Timer(Sender: TObject);
begin
  Gauge1.Progress := Gauge1.Progress+1;
  if Gauge1.Progress >= Gauge1.MaxValue then begin
    Timer1.Enabled := False;
    Self.Close;
  end;
end;

procedure TFormSplash.Image1Click(Sender: TObject);
begin
  Close;
end;

procedure TFormSplash.FormKeyPress(Sender: TObject; var Key: Char);
begin
  Close;
end;

end.
```

```
unit Fdocinfo;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics,
Controls,
  Forms, Dialogs, StdCtrls, Buttons;

type
  TFormDocInfo = class(TForm)
    Label1: TLabel;
    Edit1: TEdit;
    OKBtn: TBitBtn;
    CancelBtn: TBitBtn;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FormDocInfo: TFormDocInfo;

implementation

{$R *.DFM}

end.
```

```
unit Gomory;

interface

uses {WinCrt, }SysUtils, WinTypes, WinProcs, Messages, Classes,
StdCtrls, TokUtils;

const
  RLebar = 10;
  FloatFormat = '####0.00';
  IntegerFormat = '00000';

  mstart=2;
  mmax=22;
  ncolsmax=26;
  fwt=10;
  dpt=2;
  fwi=3;
  largevalue=1.0e20;
  smallvalue=1.0e-10;

type
  mrange = 1..mmax;
  ncolsrange = 1..ncolsmax;
  matrix = array [mrange, ncolsrange] of real;
  column = array [mrange] of real;
  baseindex = array [mrange] of integer;
  row = array [ncolsrange] of real;
  rowboolean = array [ncolsrange] of boolean;
  phase = (PhaseI, PhaseII);

  TGomorySimplex=class
  private
    d: row;
    basic: baseindex;
    nonbasic: rowboolean;
    w0, z0: real;
    it: integer;
    solution, OK: boolean;
    r, s: integer;
    n1, n2, GCPlusLC, GCPlusEC: integer;
    printon: boolean;
    slack: row;
  public
    TakTerbatas: Boolean;
    a: matrix;
    b: column;
    c: row;
    Hasil: column;
    nvar, m: Integer;
    GC, EC, LC: integer;
    HasilMemo: TMemo;
    HasilDetilMemo: TMemo;
    procedure Echo(vstring: string);
    procedure ClearOutput;
    procedure EchoLn(vstring: string);
```

```
    constructor Create;
    destructor Destroy;
    procedure SimplexMod(p:Phase);
    procedure SimplexProcess;
    function IntegerSolution: boolean;
    procedure AddConstraint(k:integer);
    function PosFrac(x: real): real;
    procedure OutputTableau(p: phase);
    procedure CompleteTableau;
    procedure Initialise;
    procedure InputData;
  end;

implementation

function FormatReal(vData: Extended; lebar: Integer): string;
var vstr: string;
begin
  vstr := FormatFloat(FloatFormat, vData);
  result := LPad(vstr, lebar);
end;

function FormatInteger(vData: Longint; lebar: Integer): string;
begin
  result := RPad(IntToStr(vData), lebar);
end;

constructor TGomorySimplex.Create;
begin
  inherited Create;
end;

destructor TGomorySimplex.Destroy;
begin
  inherited Destroy;
end;

procedure TGomorySimplex.InputData;
var i, j, k: integer;
begin
  printon := True;

  nvar := 3;
  m := 3;
  {ncols := 6;}
  c[1] := -9;
  c[2] := -10;
  c[3] := -15;
  a[1, 1] := 1;
  a[1, 2] := 2;
  a[1, 3] := 5;
  a[2, 1] := 2;
  a[2, 2] := 3;
  a[2, 3] := 3;
  a[3, 1] := 1;
  a[3, 2] := 1;
```

```
  a[3, 3] := 2;
  b[1] := 36;
  b[2] := 48;
  b[3] := 22;
  gc := 0;
  ec := 0;
  lc := 3;
{
  printon := True;
  a[1,1] := 3;
  a[1,2] := -4;
  a[2,1] := 1;
  a[2,2] := 2;
  b[1] := 12;
  b[2] := 8;
  c[1] := -5;
  c[2] := -2;
  GC := 0;
  EC := 0;
  LC := 2;
  m := 2;
  nvar := 2;
  }
end; {input data}

procedure TGomorySimplex.Initialise;
var i, j: Integer;
begin
  it := 0;
  z0 := 0.0;
  OK := true;
  GCPlusLC := GC+LC;
  GCPlusEC := GC+EC;
  n1 := nvar + GCPlusLC + GCPlusEC;
  n2 := nvar+GCPlusLC;
  for j:=nvar+1 to n1 do begin
    for i:=1 to m do begin
      a[i, j] := 0.0;
    end;
    c[j] := 0.0;
  end;
end;

procedure TGomorySimplex.CompleteTableau;
var i, j: integer;
  sum: real;
begin
  for i:= 1 to GC do begin
    a[i, nvar+i] := -1.0;
  end;
  for i:= 1to LC do begin
    a[GCPlusEC+i, nvar+GC+i] := 1.0;
  end;
  for i:= 1 to GCPlusEC do begin
    a[i, nvar+GCPlusLC+i] := 1.0;
  end;
```

```
  {compute initial base}
  for j:= 1 to GCPlusEC do basic[j] := nvar+GCPlusLC+j;
  for j:=1 to LC do basic[GCPlusEC+j] := nvar+GC+j;
  for j:= 1 to n1 do nonbasic[j] := true;
  for i:=1 to m do nonbasic[basic[i]]:= false;
  {compute d-values and w0}
  for j:=1 to n2 do begin
    sum := 0.0;
    for i:= 1 to GCPlusEC do sum := sum+a[i,j];
    d[j] := -sum;
  end;
  sum := 0.0;
  for j:= n2+1 to n1 do d[j] := 0;
  for i:=1 to GCPlusEC do sum := sum+b[i];
  w0 := -sum;
end; {complete tableau}

procedure TGomorySimplex.OutputTableau(p: phase);
var i, j, n: integer;
begin
  if p=PhaseI then n:=n1 else n:=n2;
  echoln('');
  echoln('ITERATION '+FormatInteger(it, 3));
  echo('   BASE VAR.          VALUE ');
  for j:= 1 to n do echo('     X'+FormatInteger(j, 4));
  echoln('');
  for i:= 1 to m do begin
    echo('       X'+FormatInteger(basic[i], fwi)+'
'+FormatReal(b[i],fwt));
    for j:=1 to n do echo(FormatReal(a[i, j], fwt));
    echoln('');
  end;
  echo('       Z         '+FormatReal(z0,fwt));
  for j:=1 to n do echo(FormatReal(c[j], fwt));
  echoln('');
  if p=PhaseI then begin
    echo('       '+FormatReal(w0, fwt));
    for j:=1 to n do echo(FormatReal(d[j], fwt));
    echoln('');
  end;
end;

function TGomorySimplex.PosFrac(x: real): real;
var y: real;
  hasil: real;
begin
  if abs(x-round(x))<smallvalue then hasil:=0.0
  else begin
    y:= abs(x-trunc(x));
    if x>=0.0 then hasil := y else hasil:= 1.0-y;
  end;
  posfrac := hasil;
end;

procedure TGomorySimplex.AddConstraint(k:integer);
var i, j: integer;
```

```
begin
  m:= m+1;
  n2:= n2+1;
  nonbasic[n2] := false;
  basic[m]:=n2;
  b[m] := -posfrac(b[k]);
  a[m, n2] := 1.0;
  c[n2] := 0.0;
  for j:=1 to n2-1 do begin
    if nonbasic[j] then a[m, j] := -posfrac(a[k,j])
    else a[m,j] := 0.0;
  end;
  for i:=1 to m-1 do a[i, n2]:=0.0;
  it := it+1;
end;{addconstraint}

function TGomorySimplex.IntegerSolution: boolean;
var i:integer; stillinteger: boolean;
begin
  i:=0;
  stillinteger:=True;
  while (i<m) and stillInteger do begin
    i:=i+1;
    if posfrac(b[i])>smallvalue then begin
      addconstraint(i);
      stillinteger:=false;
    end;
  end;
  integersolution := StillInteger;
end;

procedure TGomorySimplex.SimplexMod(p:Phase);
var n: integer;
  unbounded, nonfeasible: boolean;

  procedure NextBasicVariable(var r, s: integer; x: row);
  var i, j: integer; min: real;
  begin
    min := largevalue;
    for j:=1 to n do begin
      if nonbasic[j] then begin
        if x[j]<min then begin
          min := x[j];
          s:=j;
        end;
      end;
    end;
    solution := x[s]>-smallvalue;
    if not solution then begin
      unbounded := true;
      i:=1;
      while unbounded and(i<=m) do begin
        unbounded := a[i, s]<smallvalue;
        i:=i+1;
      end;
      if not unbounded then begin
```

```
        min := largevalue;
        for i:=1 to m do begin
          if a[i, s]>smallvalue then begin
            if b[i]/a[i, s]<min then begin
              min := b[i]/a[i,s];
              r:=i;
            end;
          end;
        end;
      end;
    end else begin
      min := largevalue;
      for i:=1 to m do begin
        if b[i]<min then begin
          min := b[i];
          r:=i;
        end;
      end;
      solution := b[r]>-smallvalue;
      if not solution then begin
        min := -largevalue;
        for j:=1 to n do begin
          if nonbasic[j] then begin
            if a[r,j]<-smallvalue then begin
              if x[j]/a[r,j]>min then begin
                min := x[j]/a[r,j];
                s:=j;
              end;
            end;
          end;
          nonfeasible := min<-largevalue+smallvalue;
        end;
      end;
    end;
    if not (solution or unbounded or nonfeasible) then
    begin
      nonbasic[basic[r]] := true;
      nonbasic[s]:=false;
      basic[r]:=s;
      echoln('');
      echoln('PIVOT IS AT ROW'+FormatInteger(r,
fwi)+'COL'+FormatInteger(s, fwi));
    end;
  end;{NextBasicVariable}

  procedure TransformTableau(r, s: integer; var x: row; var x0:
real);
  var i, j: integer;
    pivot, savec, savex: real;
    savecol: column;
  begin
    for i:=1 to m do savecol[i]:= a[i, s];
    savex:=x[s];
    pivot := a[r, s];
    b[r] := b[r]/pivot;
    for j:=1 to n do a[r, j] := a[r, j]/pivot;
```

```
    for i:=1 to m do
      if i<> r then begin
        b[i] := b[i]-savecol[i]*b[r];
        for j:=1 to n do a[i, j] := a[i, j] - savecol[i]*a[r, j];
      end;
      for j:=1 to n do x[j] := x[j]- savex*a[r, j];
      x0 := x0 - savex*b[r];
      if p=PhaseI then begin
        savec:=c[s];
        for j:=1 to n do c[j] := c[j]-savec*a[r, j];
        z0 := z0-savec*b[r];
      end;
  end; {transformtableau}

begin {SimplexMod}
  solution :=false;
  unbounded:=false;
  nonfeasible := false;
  if p=PhaseI then n:=n1 else n:=n2;
  repeat
    if printon then outputtableau(p);
    case p of
      PhaseI: nextbasicvariable(r, s, d);
      PhaseII: nextbasicvariable(r, s, c);
    end;
    if Not(solution or unbounded or nonfeasible) then begin
      case p of
        PhaseI: transformtableau(r, s, d, w0);
        PhaseII: transformtableau(r, s, c, z0);
      end;
      it := it+1;
    end;
  until solution or unbounded or nonfeasible;
  if unbounded then echoln('UNBOUNDED ')
  else if nonfeasible then echoln('NO FEASIBLE SOLUTION');
end;{SimplexMod}

procedure TGomorySimplex.SimplexProcess;
var i: integer;
begin
  PrintOn := True;
  echoln('');
  echoln('GOMORY''S METHOD');
  echoln('');
  m:=mstart;
  initialise;
  completetableau;
  if GCPlusEC=0 then echoln('THERE IS NO PHASE I')
  else begin
    echoln('PHASE I');
    Simplexmod(PhaseI);
    echoln('');
    if (abs(w0)>smallValue) or (not Solution) then begin
      OK:=False;
      echoln('PHASE I NOT COMPLETED ');
      echoln('SUM OF ARTIFICIALS '+FormatReal(w0,fwt));
```

```
    end else begin
      echoln('');
      echoln('PHASE I SUCCESSFUL');
      echoln('');
      echoln('REDUCED TABLEAU FOR PHASE II');
    end;
  end;
  if OK then begin
    Simplexmod(PhaseII);
    echoln('');
    if not Solution then echoln('PHASE II NOT COMPLETED ')
    else begin
      while not IntegerSolution or not Solution do
SimplexMod(PhaseII);
      echoln('');
      echoln('FINAL SOLUTION');
      echoln('');
      echoln('MAXIMUM of Z = '+FormatReal(z0,fwt));
      echoln('');
      echoln('CONSTRAINT   BASIS       VALUE       STATE
SLACK ');
      for i:=1 to m do begin
        slack[basic[i]] := b[i];
        Hasil[i] := 0;
      end;
      for i:=1 to m do begin
        if basic[i]<nvar then begin
          Hasil[basic[i]] := b[i];
        end;
        echo (Lpad(FormatInteger(i,4),
10)+Lpad(FormatInteger(basic[i],4),10)+
          FormatReal(b[i],fwt)+'       ');
        if (i<GC) or (i>GCPlusEC) then begin
          if nonbasic[nvar+i] then begin
            echoln('BINDING  '+FormatReal(0.0, 10));
            {Hasil[i] := b[i];}
          end else begin
            echoln('SLACK    '+FormatReal(slack[nvar+i],fwt));
          end;
        end else begin
          echoln(  'EQUATION       NONE');
          {Hasil[basic[i]] := b[i];}
        end;
      end;
    end;
  end;
end;
```

```
procedure TGomorySimplex.Echo(vstring: string);
begin
  if HasilDetilMemo<>nil then begin
    with HasilDetilMemo do begin
      Lines[Lines.Count-1] := Lines[Lines.Count-1]+vString;
    end;
  end;
end;

procedure TGomorySimplex.EchoLn(vstring: string);
begin
  if HasilDetilMemo <> nil then begin
    with HasilDetilMemo do begin
      Lines[Lines.Count-1] := Lines[Lines.Count-1]+vString;
      Lines.Add('');
    end;
  end;
end;

procedure TGomorySimplex.ClearOutput;
begin
  HasilDetilMemo.Clear;
  HasilDetilMemo.Lines.Add('');
end;


end.
```

```
unit FPrintHirarki;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
  QuickRpt, Qrctrls, ExtCtrls, UDataStru, UTreeDisplay,
KFieldObjects;

type
  TFormPrintHirarki = class(TForm)
    QuickRep1: TQuickRep;
    QRBand1: TQRBand;
    QRBand2: TQRBand;
    QRBand3: TQRStringsBand;
    QRImage1: TQRImage;
    QRExpr1: TQRExpr;
    QRFooter: TQRLabel;
    QRStringsBand1: TQRStringsBand;
    QRJudul: TQRLabel;
    QRHierJudul: TQRLabel;
    procedure FormCreate(Sender: TObject);
    procedure QRStringsBand1AfterPrint(Sender: TQRCustomBand;
      BandPrinted: Boolean);
  private
    { Private declarations }
    FDataStru: TRDataStru;
    FTreeDisplay: TRTreeDisplayer;
    FTop: Integer;
    procedure PreProcess(vCrit: TKBlockObject; vLevel: Integer);
    procedure SetDataStru(const Value: TRDataStru);
  public
    { Public declarations }
    procedure Cetak(Option: Integer);
    procedure CetakHierList;
    property DataStru: TRDataStru read FDataStru write SetDataStru;
  end;

var
  FormPrintHirarki: TFormPrintHirarki;

implementation

{$R *.DFM}

const CJarakLeft = 30;
  CTinggi = 20;
  CJarak = 5;
  CBatasLeft1 = 200;
  CLebarField = 150;
  CBatasKiri = 10;
```

```
procedure TFormPrintHirarki.Cetak(Option: Integer);
begin
  if (Option = 0) or (Option=1) then begin
    QRBand3.Height := QuickRep1.Height-
(QRBand1.Height+QRBand1.Height)*2-100;
    QRImage1.Height := QRBand3.Height-2*QRImage1.Top;
    QRImage1.Width := QRBand3.Width -2*QRImage1.Left;
    QRImage1.Picture.Bitmap.Width := QRImage1.Width;
    QRImage1.Picture.Bitmap.Height := QRImage1.Height;
    QRImage1.Picture.Bitmap.Canvas.Brush.Color := clWhite;
    QRImage1.Canvas.Brush.Style := bsSolid;
    QRImage1.Canvas.FillRect(Rect(0, 0, QRImage1.Width,
QRImage1.Height));
    if Option = 0 then begin
      FTreeDisplay.Canvas := QRImage1.Canvas;
      FTreeDisplay.Draw;
    end else begin
      CetakHierList;
    end;
    QuickRep1.Preview;
  end;
end;

procedure TFormPrintHirarki.FormCreate(Sender: TObject);
begin
  FTreeDisplay := TRTreeDisplayer.Create;
  //FTreeDisplay.Canvas := QRImage1.Canvas;
end;

procedure TFormPrintHirarki.SetDataStru(const Value: TRDataStru);
begin
  FDataStru := Value;
  FTreeDisplay.DataStru := FDataStru;
end;

procedure TFormPrintHirarki.QRStringsBand1AfterPrint(Sender:
TQRCustomBand;
  BandPrinted: Boolean);
begin
  QuickRep1.NewPage;
end;

procedure TFormPrintHirarki.CetakHierList;
var vTeks: string;
  xRect: TRect;
begin
  FTop := 20;

  vTeks := 'Kriteria';
  xRect.Left := CBatasKiri;
  xRect.Top := FTop;
  xRect.Right := CBatasLeft1;
  xRect.Bottom := xRect.Top+CTinggi;
  QRImage1.Canvas.Pen.Style := psSolid;
  QRImage1.Canvas.Rectangle(xRect.Left-1, xRect.Top-1,
xRect.Right+1, xRect.Bottom+1);
```

```
  QRImage1.Canvas.TextRect(xRect, xRect.Left+2, xRect.Top+2, vTeks);

  vTeks := 'Bobot Global';
  xRect.Left := xRect.Right+CJarak;
  xRect.Right := xRect.Left+CLebarField;
  QRImage1.Canvas.Rectangle(xRect.Left-1, xRect.Top-1,
xRect.Right+1, xRect.Bottom+1);
  QRImage1.Canvas.TextRect(xRect, xRect.Left+2, xRect.Top+2, vTeks);

  vTeks := 'Bobot Lokal';
  xRect.Left := xRect.Right+CJarak;
  xRect.Right := xRect.Left+CLebarField;
  QRImage1.Canvas.Rectangle(xRect.Left-1, xRect.Top-1,
xRect.Right+1, xRect.Bottom+1);
  QRImage1.Canvas.TextRect(xRect, xRect.Left+2, xRect.Top+2, vTeks);
  FTop := xRect.Top+CTinggi+CJarak;
  FDataStru.TraceTree(nil, PreProcess, nil, True);
end;

procedure TFormPrintHirarki.PreProcess(vCrit: TKBlockObject;
  vLevel: Integer);
var
  xRect: TRect;
  vTeks: String;
begin
  //if vCrit.FieldCount>0 then begin
    if vCrit.OwnerField.Owner.OwnerFIeld = nil then begin
      vTeks := 'Goal';
    end else begin
      vTeks := vCrit.OwnerField.Owner.OwnerFIeld.FieldName;
    end;
    xRect.Left := CJarakLeft*vLevel+CBatasKiri;
    xRect.Top := FTop;
    xRect.Right := CBatasLeft1;
    xRect.Bottom := xRect.Top+CTinggi;
    QRImage1.Canvas.Pen.Style := psSolid;
    QRImage1.Canvas.Rectangle(xRect.Left-1, xRect.Top-1,
xRect.Right+1, xRect.Bottom+1);
    QRImage1.Canvas.TextRect(xRect, xRect.Left+2, xRect.Top+2,
vTeks);

    vTeks := FormatFLoat('0.####', vCrit.TempReal3);
    xRect.Left := xRect.Right+CJarak;
    xRect.Right := xRect.Left+CLebarField;
    QRImage1.Canvas.Rectangle(xRect.Left-1, xRect.Top-1,
xRect.Right+1, xRect.Bottom+1);
    QRImage1.Canvas.TextRect(xRect, xRect.Left+2, xRect.Top+2,
vTeks);

    vTeks := FormatFLoat('0.####', vCrit.TempReal4);
    xRect.Left := xRect.Right+CJarak;
    xRect.Right := xRect.Left+CLebarField;
    QRImage1.Canvas.Rectangle(xRect.Left-1, xRect.Top-1,
xRect.Right+1, xRect.Bottom+1);
    QRImage1.Canvas.TextRect(xRect, xRect.Left+2, xRect.Top+2,
vTeks);
```

```
    FTop := xRect.Top+CTinggi+CJarak;
  //end;
end;

end.
```